VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV MIKROELEKTRONIKY
DEPARTMENT OF MICROELECTRONICS

IMPLEMENTACE PŘIJÍMAČE A VYSÍLAČE PROTOKOLU RMAP DO FPGA
FPGA IMPLEMENTATION OF RMAP INITIATOR AND TARGET

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Ondřej Walletzký**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Vojtěch Dvořák**
SUPERVISOR

BRNO 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Mikroelektronika**
Ústav mikroelektroniky

Student: Bc. Ondřej Walletzký ***ID:*** 146998
Ročník: 2 ***Akademický rok:*** 2016/17

NÁZEV TÉMATU:

Implementace přijímače a vysílače protokolu RMAP do FPGA

POKYNY PRO VYPRACOVÁNÍ:

Prostudujte specifikaci standardu RMAP (nadstavba sítě SpaceWire) a navrhněte verifikační model vysílače a přijímače paketů v jazyce SystemVerilog. Proveďte implementaci vysílače i přijímače paketů v jazyce VHDL, verifikujte funkci obou částí proti navrženým verifikačním modelům a návrh otestujte v cílovém obvodu FPGA.

DOPORUČENÁ LITERATURA:

Podle pokynů vedoucího práce.

Termín zadání: 6.2.2017 ***Termín odevzdání:*** 25.5.2017

Vedoucí práce: Ing. Vojtěch Dvořák
Konzultant:

doc. Ing. Lukáš Fujcik, Ph.D.
předseda oborové rady

ABSTRAKT

Práce se zabývá návrhem a implementací řadičů protokolu RMAP používaného pro přístup do paměti mezi koncovými uzly sítě SpaceWire. V teoretické části seznamuje se sítí SpaceWire, poté podrobně popisuje protokol RMAP a sběrnicové rozhraní AMBA AHB. Praktická část se věnuje návrhu architektury řadičů zmíněného protokolu na základě standardů protokolu RMAP a sběrnice AMBA AHB. Na základě navržené architektury se pak věnuje návrhu dílčích bloků. Následně popisuje použité metody verifikace navržených řadičů a jejich testování v cílovém obvodu FPGA. Nakonec analyzuje maximální frekvenci řadičů a jejich požadavky na zdroje cílového obvodu FPGA na základě odhadů syntézy.

KLÍČOVÁ SLOVA

RMAP, SpaceWire, iniciátor, cílový uzel, VHDL, SystemVerilog, AMBA, AHB, DMA, verifikace

ABSTRACT

The thesis deals with design and implementation of controllers for the RMAP protocol, which is used by SpaceWire network endpoints to access memory contents of another endpoint. The theoretical research introduces concepts of the SpaceWire network, then describes the RMAP protocol and the AMBA AHB bus interface in detail. The practical part of this thesis then uses this information to design and implement controllers for the RMAP protocol. It first defines an architecture of these controllers, then describes design of individual blocks based on this architecture. As a next step, the thesis describes methods used to verify designed controllers and to test these controllers in an FPGA chip. Finally, an analysis of maximum frequency and usage of FPGA resources is done based on estimates provided by the synthesis tool.

KEYWORDS

RMAP, SpaceWire, Initiator, Target, VHDL, SystemVerilog, AMBA, AHB, DMA, verification

WALLETZKÝ, O. *Implementace přijímače a vysílače protokolu RMAP do FPGA.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií. Ústav mikroelektroniky, 2017. 64 s., 9 s. příloh. Diplomová práce. Vedoucí diplomové práce Ing. Vojtěch Dvořák.

Prohlášení

Prohlašuji, že svoji diplomovou práci na téma Implementace přijímače a vysílače protokolu RMAP do FPGA jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne 25. května 2017 .....

podpis autora

Poděkování

Děkuji vedoucímu diplomové práce Ing. Vojtěchu Dvořákovi za metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování práce.

V Brně dne 25. května 2017 .....

podpis autora

SIX research centre
sensor, information and communication systems

Faculty of Electrical Engineering
and Communication

Brno University of Technology
Technicka 12, CZ-61600 Brno, Czechia

http://www.six.feec.vutbr.cz

Experimentální část této diplomové práce byla podpořena výzkumnou infrastrukturou vybudovanou v rámci projektu CZ.1.05/2.1.00/03.0072
Centrum senzorických, informačních a komunikačních systémů (SIX)
operačního programu Výzkum a vývoj pro inovace.

OBSAH

Úvod **11**

1 SpaceWire **12**

1.1 Fyzické médium 12

1.2 Linková vrstva 13

1.3 Síťová architektura 14

2 Protokol RMAP **17**

2.1 Pole použitá v paketech protokolu RMAP 17

2.2 Operace obecně 21

2.3 Operace zápisu 21

2.4 Operace čtení 24

2.5 Operace RMW 25

2.6 Chyby a stavové kódy 27

3 Sběrnice AMBA AHB **29**

3.1 Přehled 29

3.2 Standardní přenos 30

3.3 Přenos s přerušením 32

3.4 Arbitrace 33

3.5 Endianita 34

4 Návrh verifikačních modelů **35**

4.1 Iniciátor 36

4.2 Cílový uzel 38

5 Návrh řadičů protokolu RMAP **40**

5.1 Návrh řadiče iniciátoru 40

5.2 Návrh řadiče cílového uzlu 50

6 Ověření funkce a implementace **55**

6.1 Cílené testy 55

6.2 Náhodně generované testy 55

6.3 Testování v obvodu FPGA 57

6.4 Implementace 59

7 Závěr **60**

Literatura **61**

Seznam symbolů, veličin a zkratek **62**

Seznam příloh **64**

A Chyby paketů a reakce RMAP modulů pro operaci zápisu ..... **i**

B Chyby paketů a reakce RMAP modulů pro operaci čtení ..... **iii**

C Chyby paketů a reakce RMAP modulů pro operaci RMW ..... **iv**

D Řadič vysílače iniciátoru ..... **vi**

E Řadič přijímače iniciátoru ..... **vii**

F Řadič vysílače cílového uzlu ..... **viii**

G Řadič přijímače cílového uzlu ..... **ix**

SEZNAM OBRÁZKŮ

Obr. 1.1: Konektor sítě SpaceWire (zástrčka, pohled zepředu) (převzato z [2]) 12
Obr. 1.2: Data-Strobe kódování (převzato z [2]) 13
Obr. 1.3: Ilustrační příklad topologie sítě SpaceWire (převzato z [2]) 14
Obr. 1.4: Struktura paketu v síti SpaceWire (převzato z [2]) 14
Obr. 1.5: Adaptivní skupinové směrování (převzato z [2]) 16
Obr. 2.1: Pole instrukce (převzato z [4]) 17
Obr. 2.2: LFSR implementující polynom $x^8 + x^2 + x + 1$ (Galoisova varianta) 20
Obr. 2.3: Struktura paketu příkazu pro zápis (převzato z [4]) 22
Obr. 2.4: Struktura paketu odpovědi pro zápis (převzato z [4]) 23
Obr. 2.5: Sekvence událostí při provádění operace zápisu (převzato z [4]) 23
Obr. 2.6: Struktura paketu příkazu pro čtení (převzato z [4]) 24
Obr. 2.7: Struktura paketu odpovědi pro čtení (převzato z [4]) 24
Obr. 2.8: Sekvence událostí při provádění operace čtení (převzato z [4]) 25
Obr. 2.9: Struktura paketu příkazu pro operaci RMW (převzato z [4]) 26
Obr. 2.10: Struktura paketu odpovědi pro operaci RMW (převzato z [4]) 26
Obr. 2.11: Sekvence událostí při provádění operace RMW (převzato z [4]) 27
Obr. 3.1: Transakce s čekacími cykly (převzato z [5]) 30
Obr. 3.2: Příklad dávkové transakce o nedefinované délce (převzato z [5]) 31
Obr. 3.3: Přerušení transakce (převzato z [5]) 32
Obr. 3.4: Přidělení přístupu ke sběrnici (převzato z [5]) 33
Obr. 3.5: Odebrání přístupu arbitrem (převzato z [5]) 33
Obr. 4.1: Topologie pro ověření verifikačních modelů iniciátoru a cílového uzlu 35
Obr. 4.2: Diagram hlavní rutiny modelu iniciátoru 37
Obr. 4.3: Diagram hlavní rutiny modelu cílového uzlu 39
Obr. 5.1: Architektura iniciátoru protokolu RMAP 40
Obr. 5.2: Struktura bloků TX (nahoře) a RX (dole) 40
Obr. 5.3: Jednotka CRC 41
Obr. 5.4: Implementace výpočtu CRC – paralelní (vlevo) a LFSR (vpravo) varianta 42
Obr. 5.5: Registr transakcí 45
Obr. 5.6: Registr transakcí – paměťová buňka 46
Obr. 5.7: Registr transakcí – členění buněk v rámci regionu 47
Obr. 5.8: Schéma řadiče DMA vysílače 48
Obr. 5.9: Řadič DMA přijímače 49
Obr. 5.10: Architektura cílového uzlu protokolu RMAP 50
Obr. 5.11: Řadič DMA cílového uzlu 54
Obr. 6.1: Verifikační prostředí pro testování řadiče iniciátoru 55
Obr. 6.2: Verifikační prostředí pro testování řadiče cílového uzlu 57
Obr. 6.3: Testovací obvod pro řadič iniciátoru 57
Obr. 6.4: Testovací obvod pro řadič cílového uzlu 58
Obr. 6.5: Testovací prostředí 58

SEZNAM TABULEK

Tab. 1.1: Příklad směrovací tabulky směrovače sítě SpaceWire (převzato z [2]) 15
Tab. 1.2: Adresová mapa směrovače sítě SpaceWire (převzato z [2]) 15
Tab. 2.1: Typ paketu v protokolu RMAP (převzato z [4]) 18
Tab. 2.2: Možné kombinace bitového pole pro typ příkazu (převzato z [4]) 18
Tab. 2.3: Počet bajtů adresy v odpovědi (převzato z [4]) 19
Tab. 2.4: Chybové a stavové kódy definované v protokolu RMAP (převzato z [4]) 28
Tab. 3.1: Seznam signálů AHB sběrnice (převzato z [5]) 29
Tab. 3.2: Hodnoty signálu HTRANS (převzato z [5]) 30
Tab. 3.3: Typy transakcí (převzato z [5]) 31
Tab. 3.4: Příklad inkrementujicí adresy bez přetečení a s přetečením 31
Tab. 3.5: Hodnoty signálu HRESP (převzato z [5]) 32
Tab. 3.6: Řazení bajtů na datové sběrnici pro různé typy endianity, viz [5] 34
Tab. 5.1: Princip replikace datových bajtů pro 64bitovou datovou sběrnici 50
Tab. 6.1: Parametry řadičů 59
Tab. 6.2: Odhad využití zdrojů FPGA a maximální frekvence 59

ÚVOD

Vesmírné sondy a zařízení hrají klíčovou roli v dnešním světě. Slouží nejen jako nástroj vědců prozkoumávajících vesmír, ale například také k testování chování materiálů a organismů za podmínek na povrchu naší planety nedosažitelných. I dnes široce používaný navigační systém GPS Spojených států a projekt Galileo Evropské vesmírné agentury využívá specializovaných satelitů obíhajících kolem Země.

Rostoucí komplexita vesmírných zařízení, která mohou obsahovat velké množství senzorů pro telemetrii, jednotky zpracovávající tato data a komunikační zařízení, klade vysoké nároky na přenos dat mezi jednotlivými moduly v podmínkách, které jsou vůči elektronice velmi nehostinné.

Evropská vesmírná agentura proto v roce 2003 definovala standard rozhraní pro komunikaci mezi jednotlivými moduly, které nazvala SpaceWire. Toto rozhraní má zajistit spolehlivou komunikaci mezi dílčími částmi vesmírného zařízení s vysokou datovou propustností. Pro toto rozhraní pak ESA pomocí standardů definovala několik komunikačních protokolů, které využívají SpaceWire rozhraní. Jedním z nich je protokol pro vzdálený přístup do paměti mezi koncovými uzly sítě SpaceWire, tzv. Remote Memory Access Protocol, zkráceně také RMAP.

Cílem diplomové práce je nejprve nastudovat standard protokolu RMAP. Na základě získaných informací následuje návrh a implementace verifikačních modelů pro řadiče tohoto protokolu. Poté se práce dále zabývá návrhem a implementací reálných řadičů protokolu RMAP, jejich funkční verifikací, a nakonec testováním v cílovém obvodu FPGA vůči referenčnímu modulu.

1 SPACEWIRE

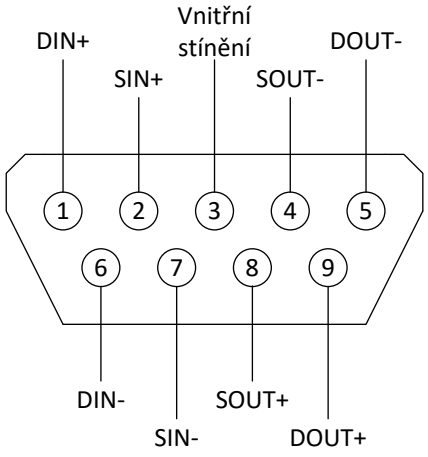
SpaceWire je komunikační rozhraní standardizované Evropskou vesmírnou agenturou. Bylo navrženo primárně pro použití v kosmickém průmyslu, kde jsou na elektronická zařízení kladeny velmi vysoké nároky na spolehlivost. SpaceWire je navržen s ohledem na spolehlivost přenosu dat a vysokou datovou propustnost od 2 do 200 Mb/s. [1]

Rozhraní vychází ze standardu IEEE 1355-1995 pro rozhraní HIC. Od tohoto rozhraní se liší v několika aspektech (viz [2]):

- Kabely a konektory byly specifikovány pro použití v kosmickém průmyslu.
- HIC používá PECL logiku pro přenos signálů. SpaceWire ji nahrazuje LVDS pro lepší elektromagnetickou kompatibilitu, větší spolehlivost a lepší kompatibilitu se současnými technologiemi výroby integrovaných obvodů.
- SpaceWire specifikuje vyšší nároky na časování signálů.
- Chování rozhraní od linkové vrstvy výše SpaceWire pozměňuje, případně upřesňuje.

1.1 Fyzické médium

Rozhraní se skládá z 8, v případě vedení kabelem 9, vodičů. Jedná se o 4 páry vodičů, které vedou diferenciální signál, 9. vodič je použit v kabelu a jedná se o stínění. Každý pár vodičů tvoří v kabelu kroucený pár obklopený stíněním. Kolem všech stíněných párů je dále společné stínění. Kabel používá pro připojení k zařízení 9pinový konektor typu D s malými rozměry, viz Obr. 1.1.



Obr. 1.1: Konektor sítě SpaceWire (zástrčka, pohled zepředu) (převzato z [2])

Ze 4 párů vodičů jsou 2 určeny pro vysílání a 2 pro příjem dat. Každá z dvojic přenáší data spolu s hodinovým signálem, přičemž je přenos zakódován pomocí tzv. Data-Strobe kódování. Při tomto způsobu kódování se pro přenos používají 2 signály:

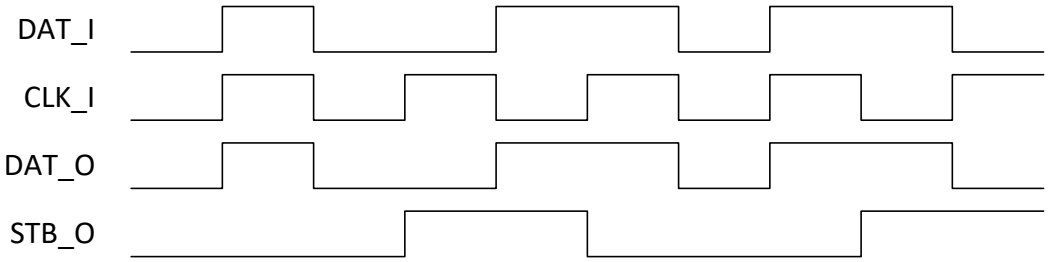
- Data – reprezentuje přenášená data v nezměněné formě
- Strobe – do tohoto signálu je zakódován hodinový signál

Data-Strobe kódování má tu vlastnost, že se v čase mění pouze jeden ze signálů, což zlepšuje vlastnosti přenosu dat z hlediska elektromagnetické kompatibility. Mějme vstupní signály DAT_I a CLK_I a výstupní signály DAT_O a STB_O. Potom dle [2] platí

$$DAT\_O = DAT\_I, \tag{1.1}$$

$$STB\_O = DAT\_I \oplus CLK\_I, \tag{1.2}$$

kde DAT_I a DAT_O reprezentují přenášená data, CLK_I je hodinový signál přenosu a STB_O reprezentuje signál Strobe. Průběhy signálů DAT_O a STB_O v závislosti na signálech DAT_I a CLK_I jsou znázorněny na Obr. 1.2.



Obr. 1.2: Data-Strobe kódování (převzato z [2])

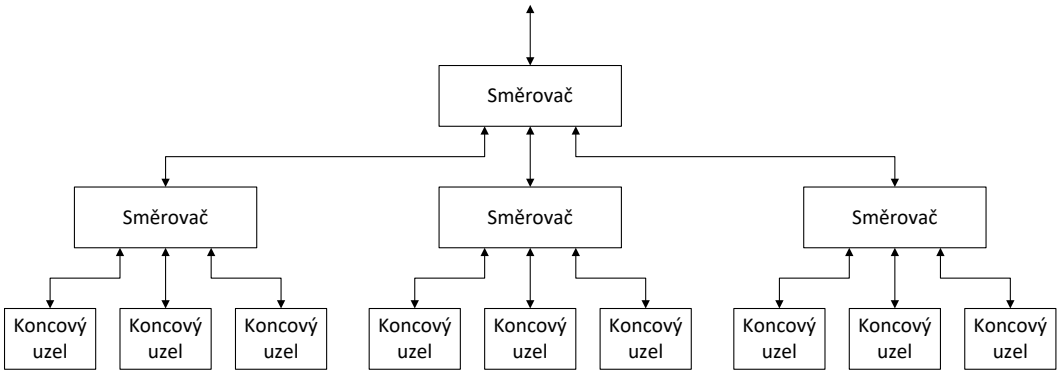
1.2 Linková vrstva

Na úrovni přenosu znaků přidává SpaceWire časové kódy, aby bylo možno synchronizovat systémový čas po celé síti. Pro kontrolu integrity dat používá pro každý znak paritní bit. Rozhraní je navrženo tak, aby po spuštění neustále vysílalo data. Pokud zrovna zařízení neposílá žádná data, je vysílán speciální znak NULL. Tím je dosažena schopnost rozhraní detekovat případné rozpojení, přičemž detekční doba je stanovena na 850 ns. [2]

Řízení toku dat je v případě sítě SpaceWire prováděno pomocí tzv. žetonů. Může-li přijímač přijmout alespoň 8 znaků, zašle žeton vysílači. Vysílač tak ví, že může přijímači zaslat až $8n$ znaků, kde n je počet přijatých žetonů. [2]

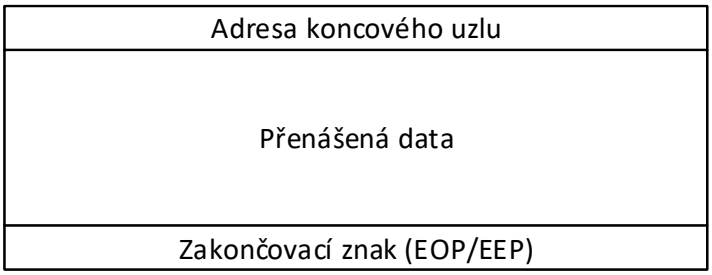
1.3 Síťová architektura

Jednotlivá SpaceWire zařízení lze propojit přímo mezi sebou. SpaceWire však podporuje také propojení v síti prostřednictvím směrovačů, viz Obr. 1.3.



Obr. 1.3: Ilustrační příklad topologie sítě SpaceWire (převzato z [2])

Síť může mít libovolnou topologii. Data jsou přenášena po paketech, které mají strukturu znázorněnou na Obr. 1.4.



Obr. 1.4: Struktura paketu v síti SpaceWire (převzato z [2])

První částí je cílová adresa paketu. Po ní následují přenášená data a konec paketu označuje speciální znak EOP, případně EEP pro indikování chyby při přenosu.

Standard rozhraní umožňuje použití tzv. Wormhole směrování, kdy směrovač okamžitě po přijetí adresy paketu identifikuje příslušný výstupní port a je-li je tento port volný, přesměruje do něj paket bez dalšího ukládání. Přitom označí výstupní port jako obsazený a neuvolní ho, dokud směrovačem neprojde celý paket. [2]

Standard sítě SpaceWire definuje několik způsobů adresování paketu. Ve všech případech však platí, že za adresu se vždy považuje první znak paketu.

1. **Adresování cestou (Path Addressing)** – adresa obsahuje čísla výstupních portů jednotlivých směrovačů po cestě k cílovému zařízení. Tento způsob adresování je nejméně náročný z hlediska množství hradel potřebných k implementaci a směrovače jsou proto jednoduché. Na druhou stranu tento způsob může vyžadovat velmi dlouhou adresu, o jejíž generování se stará zařízení odesílající paket. Při tomto způsobu adresování je v každém směrovači adresa zkrácena o jeden znak. [2]

2. **Adresování prostřednictvím logické adresy (Logical Addressing)** – při použití tohoto způsobu adresování paketů má každý cílový uzel svůj unikátní identifikátor, který reprezentuje jeho umístění v síti. Toto řešení je z hlediska složitosti směrovačů komplexnější, protože vyžaduje směrovací tabulku, která může být v případě rozsáhlejších sítí velmi velká. Tato tabulka, která se nachází ve všech směrovačích, slouží k překladu logické adresy na konkrétní výstupní port z pohledu příslušného směrovače. Pro tento způsob adresování zůstává adresa zachována po celou cestu k cíli. Příklad směrovací tabulky znázorňuje Tab. 1.1. [2]
3. **Regionální adresování (Regional Logical Addressing)** – jedná se o kombinaci adresování cestou a logické adresy, kdy je síť rozdělena do tzv. regionů. Tyto regiony jsou adresovány pomocí cesty. V rámci regionu je pak použita logická adresa. [2]
4. **Adresování pomocí intervalů (Interval Labelling)** – toto adresování je založeno na metodě logické adresy. Ke každému portu směrovače je ve směrovací tabulce přiřazen určitý rozsah adres, což umožňuje zjednodušit směrovací tabulku. [2]

Tab. 1.1: Příklad směrovací tabulky směrovače sítě SpaceWire (převzato z [2])

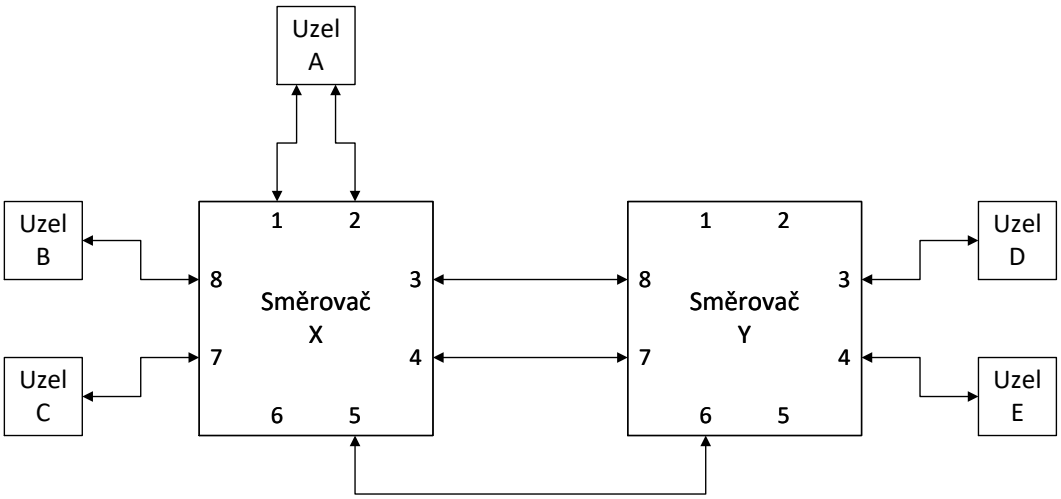
Logická adresa	Fyzický výstupní port
1	8
2	1
3	3
4	1
…	…

SpaceWire specifikuje maximální počet portů směrovače na 32. Dále také definuje rozsah adres pro fyzické porty směrovače, pro logickou adresu a pro konfigurační port směrovače, viz Tab. 1.2. [2]

Tab. 1.2: Adresová mapa směrovače sítě SpaceWire (převzato z [2])

Adresový rozsah	Význam	Mazání hlavičky
0 (00_{16})	Konfigurační port směrovače	Ano
1–31 (01_{16}–$1F_{16}$)	Fyzické výstupní porty	Ano
32–254 (20_{16}–FE_{16})	Logické adresy mapované na fyzické porty	Volitelné pro každý výstupní port. Hlavička je mazána, pokud port vede do jiného adresového regionu, anebo v posledním směrovači.
255 (FF_{16})	Stejný význam jako logické adresy, ale tato je rezervována pro použití v budoucnu.	Volitelné pro každý výstupní port. Hlavička je mazána, pokud port vede do jiného adresového regionu, anebo v posledním směrovači.

Standard také definuje tzv. Adaptivní skupinové směrování, které umožňuje paralelně přenést transakce z více zdrojů do jedné podsítě. Pro tento způsob přenosu mají směrovače více vzájemně kompatibilních portů, které jsou připojené do stejného cílového směrovače, a sledují obsazenost jednotlivých portů. Pokud více zařízení chce vysílat do stejné podsítě, směrovač může využít dostupné porty a přiřadit je jednotlivým vysílačům. Příklad topologie pro tento způsob směrování lze vidět na Obr. 1.5. [2]



Obr. 1.5: Adaptivní skupinové směrování (převzato z [2])

2 PROTOKOL RMAP

Remote Memory Access Protocol, zkráceně RMAP, je jedním z protokolů navržených pro použití s rozhraním SpaceWire. Tento protokol je specifikován standardem [4] a jeho cílem je podpora přístupu do paměti v síti SpaceWire. Tento protokol lze použít ke konfiguraci sítě SpaceWire, řízení jednotlivých uzlů a přenášení dat mezi nimi. Protokol RMAP může běžet zároveň s ostatními protokoly definovanými pro rozhraní SpaceWire. [3]

RMAP může být například využit ke konfiguraci směrovačů sítě SpaceWire nastavováním jejich pracovních parametrů a směrovacích tabulek nebo konfigurovat jednotlivé uzly v síti, například jejich přenosovou rychlost. Protože se jedná o protokol navržený pro práci s pamětí, může být použit pro práci s čímkoli, k čemu lze přistoupit prostřednictvím adresy v adresovém prostoru cílového zařízení. [3]

2.1 Pole použitá v paketech protokolu RMAP

Pro snazší pochopení jednotlivých operací protokolu jsou v této sekci uvedeny jednotlivé části paketů definovaných pro protokol RMAP a jejich popis. Po této sekci následuje popis jednotlivých operací včetně struktur příslušných paketů.

2.1.1 Adresa koncového uzlu sítě SpaceWire

Obsahuje adresu koncového uzlu v síti SpaceWire. Dovoleno je adresování pomocí cesty nebo regionální adresování. Tato adresa nesmí být použita, pokud je pro adresování použita pouze logická adresa cílového uzlu. Během cesty napříč sítí je postupně odstraněna jednotlivými směrovači. [4]

2.1.2 Logická adresa cílového uzlu

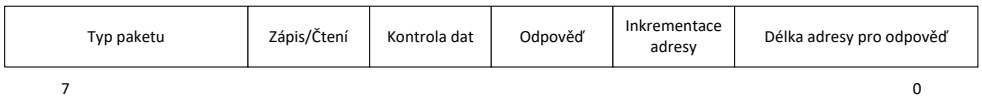
Obsahuje jeden bajt logické adresy cílového uzlu. Ten může mít více logických adres. Pokud cílový uzel nerozlišuje žádnou logickou adresu, může iniciátor nastavit toto pole na výchozí hodnotu. Tato hodnota je podle standardu FE_{16}. [4]

2.1.3 Identifikátor protokolu

Jeden bajt, který udává, kterému protokolu daný paket přísluší. Identifikátor protokolu RMAP má hodnotu 01_{16}. [4]

2.1.4 Pole instrukce

Pole instrukce je pole o velikosti jednoho bajtu, které obsahuje typ paketu, typ příkazu a délku adresy pro odeslání odpovědi (viz [4]). Složení pole instrukce je znázorněno na Obr. 2.1.



Obr. 2.1: Pole instrukce (převzato z [4])

Typ paketu

Jedná se o pole šířky 2 bitů, které specifikuje typ paketu. Významy jednotlivých kombinací jsou uvedeny v Tab. 2.1.

Tab. 2.1: Typ paketu v protokolu RMAP (převzato z [4])

Binární hodnota	Význam
00	Odpověď
01	Příkaz
10, 11	Rezervováno

Typ příkazu

Jedná se o 4bitové pole, použité v paketech příkazu i odpovědi, a udává typ operace. Udává, zda se jedná o operaci zápisu nebo čtení, zda má být provedena kontrola integrity dat, zda je vyžadována odpověď a jestli má být adresa při přístupu do paměti postupně inkrementována. Možné kombinace tohoto bitového pole jsou shrnuty v Tab. 2.2.

Tab. 2.2: Možné kombinace bitového pole pro typ příkazu (převzato z [4])

Bit 5	Bit 4	Bit 3	Bit 2	Číslo bitu v poli instrukce
Zápis/ Čtení	Kontrola dat	Odpověď	Inkrementace adresy	Funkce
0	0	0	0	Neplatné
0	0	0	1	Neplatné
0	0	1	0	Čtení, jedna adresa
0	0	1	1	Čtení, inkrementace adresy
0	1	0	0	Neplatné
0	1	0	1	Neplatné
0	1	1	0	Neplatné
0	1	1	1	RMW, inkrementace adresy
1	0	0	0	Zápis, jedna adresa, bez odpovědi a kontroly dat
1	0	0	1	Zápis, inkrementace adresy, bez odpovědi a kontroly dat
1	0	1	0	Zápis, jedna adresa, s odpovědí, bez kontroly dat
1	0	1	1	Zápis, inkrementace adresy, s odpovědí, bez kontroly dat
1	1	0	0	Zápis, jedna adresa, bez odpovědi, s kontrolou dat
1	1	0	1	Zápis, inkrementace adresy, bez odpovědi, s kontrolou dat
1	1	1	0	Zápis, jedna adresa, s odpovědí a kontrolou dat
1	1	1	1	Zápis, inkrementace adresy, s odpovědí a kontrolou dat

Délka adresy pro odpověď

Toto 2bitové pole udává počet bajtů adresy pro odpověď. Počet bajtů adresy odpovědi je k jednotlivým hodnotám tohoto pole přiřazen podle Tab. 2.3. Reprezentuje nejmenší potřebný počet 32bitových slov potřebných k uchování adresy odpovědi. [4]

Tab. 2.3: Počet bajtů adresy v odpovědi (převzato z [4])

Hodnota pole	Délka adresy
00_2	0 bajtů
01_2	4 bajty
10_2	8 bajtů
11_2	12 bajtů

2.1.5 Přístupový klíč

Jedná se o 8bitové pole obsahující klíč rozeznatelný aplikací cílového uzlu. Tím je možno ověřit práva přístupu k paměti cílového uzlu.

2.1.6 Adresa odpovědi

Toto pole obsahuje adresu odesílatele příkazu z pohledu cílového uzlu, aby případná odpověď dorazila, kam má. Velikost tohoto pole je dána nastavením v poli instrukce podle Tab. 2.3.

Toto pole se používá k adresování cestou nebo k regionálnímu adresování. Tato adresa není nutná, pokud je pro směrování paketu odpovědi použita logická adresa. Standard udává, že nulové bajty na začátku adresy budou ignorovány, s výjimkou případu, kdy je nastaven nenulový počet bajtů adresy a všechny tyto bajty mají hodnotu 00_{16}. V tom případě je jako adresa odpovědi poslán jeden nulový znak. To proto, aby byla platná také adresa odpovědi obsahující jeden nulový znak. [4]

2.1.7 Logická adresa iniciátoru

Toto pole o velikosti jednoho bajtu obsahuje logickou adresu iniciátoru, má-li iniciátor nějakou, jinak obsahuje hodnotu FE_{16}, což je výchozí hodnota, viz [4]. Iniciátor může mít dle [4] více logických adres.

2.1.8 Identifikátor transakce

Jedná se o 16bitové pole reprezentující unikátní identifikátor příkazu. Slouží k přiřazení odpovědi k příkazu, který odeslání odpovědi zapříčinil, tj. příkazu, ke kterému daná odpověď náleží. Identifikátor odpovědi a souvisejícího příkazu musí mít stejnou hodnotu. Jako první je odeslán bajt MSB. [4]

2.1.9 Rozšířená adresa

Pole rozšířené adresy je 8bitové pole, které slouží k rozšíření 32bitové adresy na 40 bitů. V rámci rozšířené adresy reprezentuje nejvyšších 8 bitů. [4]

2.1.10 Adresa

Pole o délce 32 bitů, které obsahuje spodních 32 bitů rozšířené adresy. Jako první je odeslán bajt MSB. [4]

2.1.11 Délka datového pole

Toto pole má délku 24 bitů a obsahuje množství posílaných dat nebo dat společně s maskou v bajtech. Jako první je odeslán bajt MSB. [4]

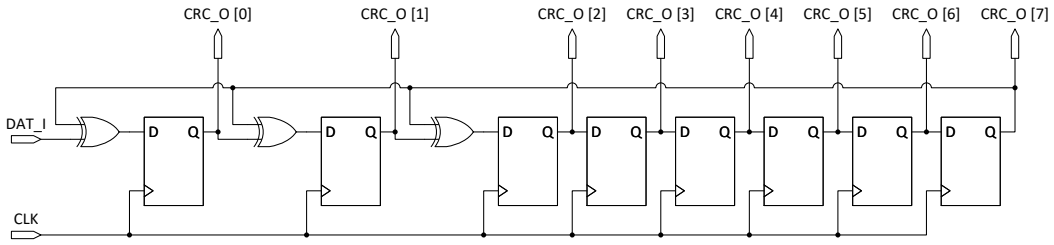
2.1.12 CRC hlavičky

Pole o velikosti jednoho bajtu, které obsahuje CRC pokrývající všechny bajty hlavičky od bajtu logické adresy po poslední bajt před CRC hlavičky. [4]

Standard protokolu RMAP, viz [4], pro výpočet CRC kódu definuje polynom

$$g(x) = x^8 + x^2 + x + 1. \tag{2.1}$$

V praxi je výpočet CRC implementován pomocí registru LFSR. Jedná se o posuvný registr, jehož výstup je lineární funkcí předchozího stavu. V případě protokolu RMAP je CRC kód počítán pomocí Galoisovy varianty LFSR, viz Obr. 2.2. Standard protokolu specifikuje, že pro tuto implementaci LFSR je počáteční hodnota registru rovna nule a že pro výpočet jsou jednotlivé bajty bitově převrácené, jinými slovy, vstupní bajt vstupuje do LFSR bitem LSB napřed. [4]



Obr. 2.2: LFSR implementující polynom $x^8 + x^2 + x + 1$ (Galoisova varianta)

2.1.13 Data

Jedná se o pole s proměnným počtem bajtů, který je upřesněn v hlavičce paketu. Toto pole obsahuje data pro zápis nebo pro čtení. [4]

2.1.14 Maska

Jedná se o pole s proměnným počtem bajtů, který je upřesněn v hlavičce paketu. Toto pole obsahuje bitovou masku přenášenou jako část RMW paketu a jeho délka se shoduje s délkou pole zaslaných dat. [4]

2.1.15 CRC dat

Pole o velikosti jednoho bajtu, které obsahuje CRC pokrývající všechny bajty mezi CRC hlavičky a CRC dat, tj. všechny bajty dat, v případě RMW paketu také všechny bajty masky. CRC dat je počítáno stejným způsobem, jakým je počítáno CRC hlavičky. [4]

2.1.16 Adresa odpovědi v síti SpaceWire

Toto pole má proměnný počet bajtů a představuje adresu odpovědi pro směrování v rámci sítě SpaceWire. Tato adresa je vytvořena z adresy odpovědi přijaté v příkazu. [4]

2.1.17 Status

Jedná se o 8bitové pole obsahující kód stavu či chyby. [4]

2.2 Operace obecně

Všechny operace definované ve standardu RMAP protokolu jsou tzv. posted operace, jinými slovy, iniciátor po odeslání příkazu nečeká na odpověď, a tím pádem může odeslat několik příkazů po sobě, než dorazí první odpověď. Proto má každý příkaz svůj identifikátor jedinečný v rámci onoho iniciátoru a logické adresy cílového uzlu. Protokol RMAP neobsahuje žádný mechanismus, který by kontroloval vypršení maximální doby pro přijetí odpovědi. Pokud je takový mechanismus žádoucí, měl by být implementován v uživatelské aplikaci. [4]

2.3 Operace zápisu

Při provádění této operace iniciátor provádí zápis nula a více bajtů dat do paměti cílového uzlu. Operace zápisu mohou být volitelně s odpovědí nebo kontrolou integrity dat před samotným zápisem do paměti. Dle standardu tedy existují celkem 4 varianty operace zápisu, viz [4].

Zápis bez odpovědi a kontroly dat

Zapisuje nula nebo více bajtů do paměti cílového uzlu. Integrita hlavičky příkazu je zkontrolována prostřednictvím CRC, ale samotná data před zápisem do paměti kontrolována nejsou. Cílový uzel neposílá iniciátoru žádnou odpověď, případný výsledek operace může iniciátor zkontrolovat přístupem do stavových registrů cílového uzlu. [4]

Tento druh zápisu se používá především pro zápis velkého množství dat, přičemž lze předpokládat, že data byla úspěšně zapsána. Příkladem může být přenos dat z kamery do vyrovnávací paměti. [4]

Zápis bez odpovědi, ale s kontrolou dat

Zapisuje nula nebo více bajtů do paměti cílového uzlu. Integrita hlavičky příkazu i samotných dat je zkontrolována prostřednictvím CRC ještě před tím, než jsou data zapsána do paměti. Je zde omezení množství dat odeslaných v rámci jednoho příkazu dané kapacitou kontrolní vyrovnávací paměti. Na druhou stranu je nepravděpodobné, že by došlo k zápisu chybných dat do paměti. [4]

Tento druh zápisu se používá pro zápis dat do citlivých bloků paměti cílového uzlu, jakými jsou například konfigurační registry, přičemž lze bezpečně předpokládat, že data byla úspěšně zapsána. Vzhledem k tomu, že není vyžadována odpověď, musí iniciátor pro zjištění výsledku operace přečíst obsah stavových registrů cílového uzlu. [4]

Zápis s odpovědí, ale bez kontroly dat

Zapisuje nula nebo více bajtů do paměti cílového uzlu. Integrita hlavičky příkazu je zkontrolována prostřednictvím CRC, ale samotná data před zápisem do paměti kontrolována nejsou. Iniciátoru je zaslána odpověď indikující výsledný stav operace. [4]

Tento druh zápisu se používá zejména v případech, kdy je potřeba zapsat velké množství dat, ale zároveň je potřeba upozornění z cílového uzlu, zda došlo k zápisu bez problému, nebo zda došlo při zápisu k chybě. [4]

Zápis s odpovědí a s kontrolou dat

Zapisuje nula nebo více bajtů do paměti cílového uzlu. Integrita hlavičky příkazu i samotných dat je zkontrolována prostřednictvím CRC ještě před tím, než jsou data zapsána do paměti. Je zde omezení množství dat odeslaných v rámci jednoho příkazu dané kapacitou kontrolní vyrovnávací paměti. Na druhou stranu je nepravděpodobné, že by došlo k zápisu chybných dat do paměti. Iniciátoru je zaslána odpověď indikující výsledný stav operace. [4]

Tento druh zápisu lze použít například pro zápis do citlivých bloků paměti, jakými jsou konfigurační registry, přičemž iniciátoru přijde odpověď s výsledkem operace zápisu. [4]

Struktura paketu příkazu je znázorněna na Obr. 2.3.

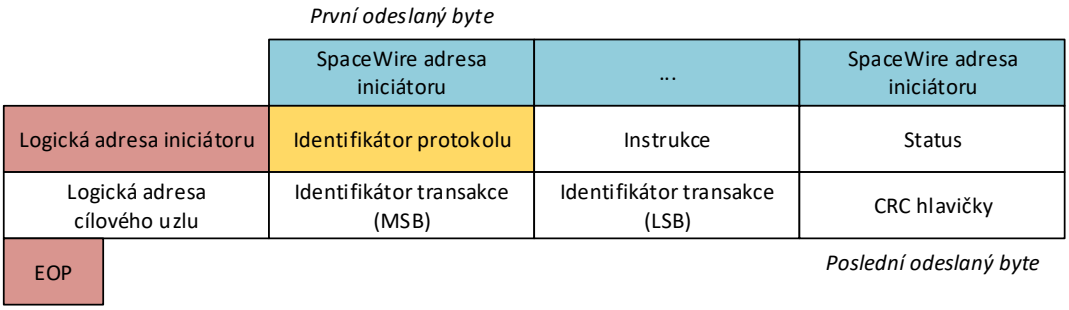
První odeslaný byte

	SpaceWire adresa cílového uzlu	…	SpaceWire adresa cílového uzlu
Logická adresa cílového uzlu	Identifikátor protokolu	Instrukce	Klíč
Adresa odpovědi	Adresa odpovědi	Adresa odpovědi	Adresa odpovědi
Adresa odpovědi	Adresa odpovědi	Adresa odpovědi	Adresa odpovědi
Adresa odpovědi	Adresa odpovědi	Adresa odpovědi	Adresa odpovědi
Logická adresa iniciátoru	Identifikátor transakce (MSB)	Identifikátor transakce (LSB)	Rozšířená adresa paměti
Adresa paměti (MSB)	Adresa paměti	Adresa paměti	Adresa paměti (LSB)
Počet bytů dat (MSB)	Počet bytů dat	Počet bytů dat (LSB)	CRC hlavičky
Data	Data	Data	Data
Data	…	…	Data
Data	CRC dat	EOP	

Poslední odeslaný byte

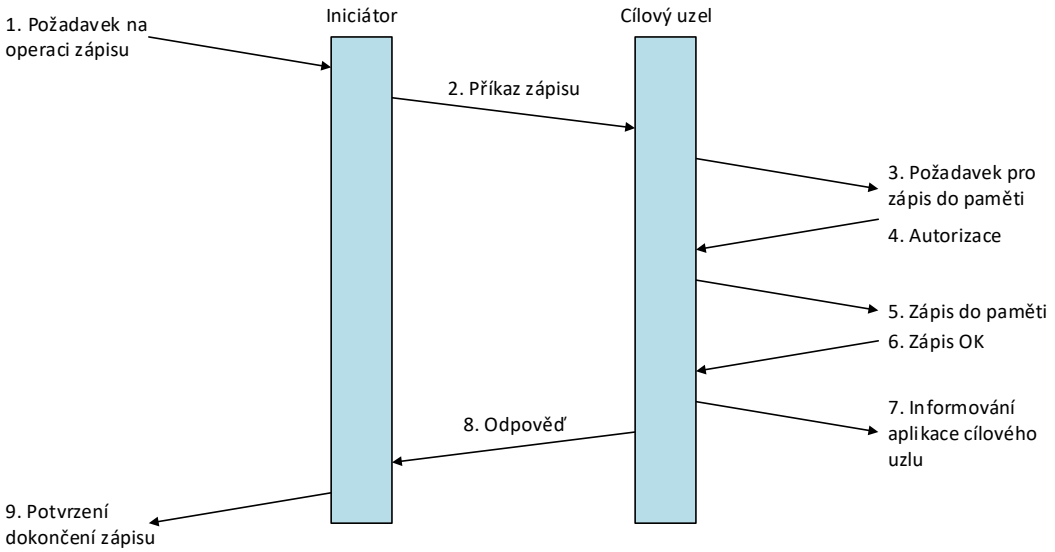
Obr. 2.3: Struktura paketu příkazu pro zápis (převzato z [4])

Struktura paketu odpovědi je zobrazena na Obr. 2.4.



Obr. 2.4: Struktura paketu odpovědi pro zápis (převzato z [4])

Obecnou sekvenci operací nutných pro úspěšný zápis lze vidět na Obr. 2.5. Prvně aplikace iniciátoru požádá svůj RMAP modul o zápis do paměti cílového uzlu. Ten po přijetí všech potřebných dat vytvoří paket příkazu a pošle ho cílovému uzlu. Ten při přijetí paketu zkontroluje jeho hlavičku, a je-li vše v pořádku, požádá o přístup k paměti pro zápis. Zde může cílový uzel odmítnout přijetí transakce např. z důvodu nesprávného klíče. Je-li přístup schválen, přejde cílový uzel do fáze zápisu. Je-li vyžadována kontrola integrity zapisovaných dat, nejdříve data zkontroluje a jsou-li v pořádku, zapíše je do paměti. Není-li vyžadována kontrola integrity dat, zápis bude proveden rovnou. Po zápisu dat upozorní cílový uzel jeho nadřazenou aplikaci, že došlo k zápisu dat. Nakonec odešle paket odpovědi, bylo-li to požadováno. [4]



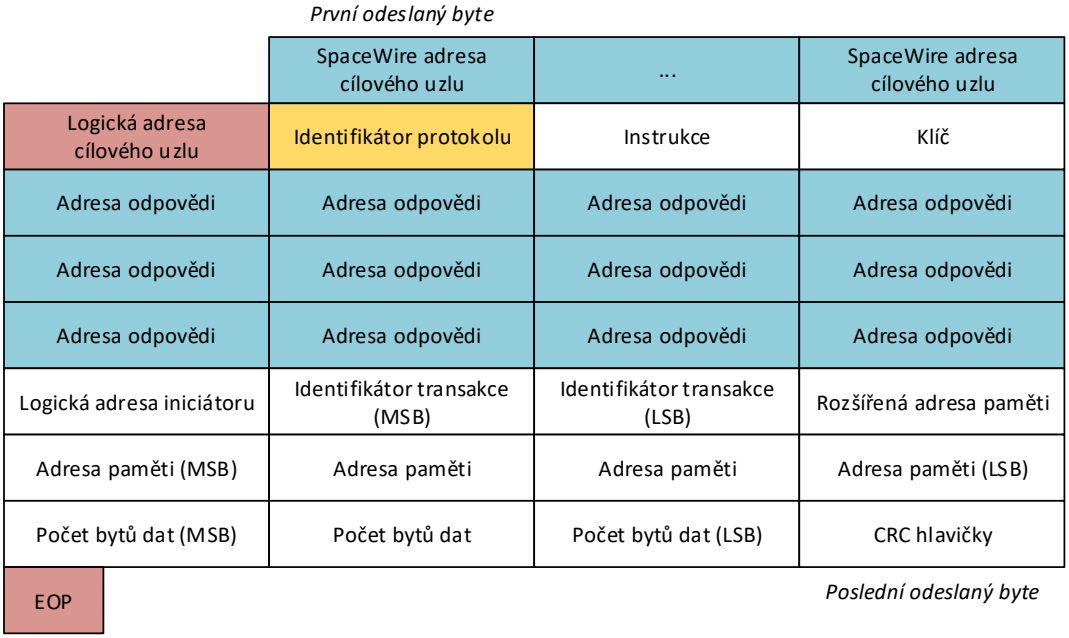
Obr. 2.5: Sekvence událostí při provádění operace zápisu (převzato z [4])

Během operace zápisu může dojít k rozličným chybám. Při některých cílový uzel zahodí paket bez odeslání odpovědi, při jiných může odpověď poslat, pokud to iniciátor v rámci příkazu požadoval. Ve všech případech však při výskytu chyby nastaví příslušný indikátor chyby ve stavovém registru. Možné chyby, které mohou nastat, a způsoby chování přijímacího modulu jsou shrnuty v Příloze A.

2.4 Operace čtení

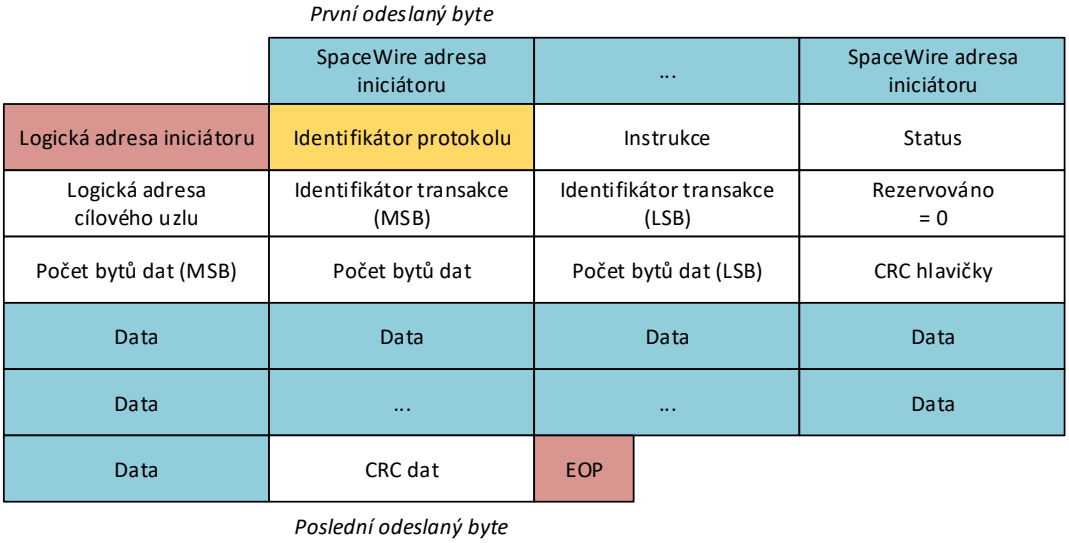
Při provádění operace čtení iniciátor čte nula nebo více bajtů ze specifikovaného bloku paměti cílového uzlu. Čtená data jsou odeslána iniciátoru jako paket odpovědi. [4]

Struktura paketu příkazu je znázorněna na Obr. 2.6.



Obr. 2.6: Struktura paketu příkazu pro čtení (převzato z [4])

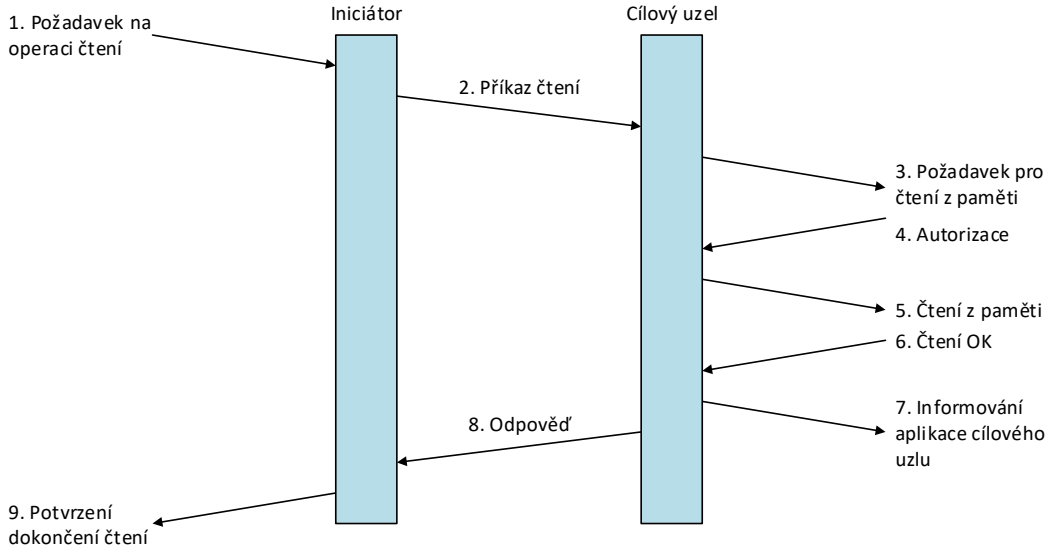
Na Obr. 2.7 lze vidět strukturu paketu odpovědi.



Obr. 2.7: Struktura paketu odpovědi pro čtení (převzato z [4])

Obecnou sekvenci operací nutných pro úspěšnou operaci čtení lze vidět na Obr. 2.8. Nejdříve aplikace iniciátoru požádá svůj RMAP modul o čtení z paměti cílového uzlu

a předá mu potřebná data. RMAP modul iniciátoru pak sestaví paket a odešle ho cílovému uzlu. Ten zkontroluje hlavičku a je-li vše v pořádku, požádá o přístup do paměti. V této fázi může být operace odmítnuta, například z důvodu nesprávného klíče pro přístup. Je-li operace schválena, dojde ke čtení vyžádaných dat a k přichystání paketu odpovědi a aplikace cílového uzlu je upozorněna, že došlo ke čtení dat. Tento paket je pak odeslán zpět iniciátoru, který zkontroluje jeho integritu. Je-li paket v pořádku, přichystá čtená data pro aplikaci a upozorní ji. [4]



Obr. 2.8: Sekvence událostí při provádění operace čtení (převzato z [4])

Stejně, jako při operaci zápisu, může během operace čtení dojít k různým chybám. Typy chyb a způsoby reakce na ně jsou shrnuty v Příloze B.

2.5 Operace RMW

Tato operace umožňuje iniciátoru zaslat cílovému uzlu data s maskou, na základě kterých cílový uzel přečte data v požadovaném bloku paměti, pozmění je a zapíše zpět na původní místo v paměti. Přitom odešle iniciátoru obsah původních dat. [4]

Obecná sekvence pro operaci RMW je znázorněna na Obr. 2.11. Aplikace iniciátoru požádá RMAP modul o RMW operaci a dodá mu všechna potřebná data. RMAP modul vytvoří odpovídající paket a pošle ho cílovému uzlu, který poté zkontroluje integritu hlavičky i dat a je-li paket v pořádku, požádá o přístup do paměti. Zde je možnost odmítnout požadovanou operaci, například z důvodu špatného přístupového klíče. Po schválení přístupu do paměti cílový uzel přečte požadovaná data, vypočítá nová data na základě dat přečtených z paměti, dat přijatých v příkazu a masky přijaté v příkazu. Tato data poté zapíše zpět do původní lokace v paměti. Po zápisu nových dat upozorní cílový uzel svoji aplikaci, že došlo k aktualizaci dat. Poté vytvoří paket odpovědi s původními daty vyčtenými z paměti před jejich přepsáním a odešle ho zpět iniciátoru. Ten zkontroluje paket odpovědi a je-li v pořádku, upozorní svoji aplikaci, že operace byla úspěšně dokončena. [4]

Struktura paketu příkazu je vyobrazena na Obr. 2.9.

První odeslaný byte

	SpaceWire adresa cílového uzlu	…	SpaceWire adresa cílového uzlu
Logická adresa cílového uzlu	Identifikátor protokolu	Instrukce	Klíč
Adresa odpovědi	Adresa odpovědi	Adresa odpovědi	Adresa odpovědi
Adresa odpovědi	Adresa odpovědi	Adresa odpovědi	Adresa odpovědi
Adresa odpovědi	Adresa odpovědi	Adresa odpovědi	Adresa odpovědi
Logická adresa iniciátoru	Identifikátor transakce (MSB)	Identifikátor transakce (LSB)	Rozšířená adresa paměti
Adresa paměti (MSB)	Adresa paměti	Adresa paměti	Adresa paměti (LSB)
Počet bytů dat (MSB) = 0	Počet bytů dat = 0	Počet bytů dat (LSB) = 0, 2, 4, 6, 8	CRC hlavičky
Data (MSB)	Data	Data	Data (LSB)
Maska (MSB)	Maska	Maska	Maska (LSB)
CRC dat	EOP		

Poslední odeslaný byte

Obr. 2.9: Struktura paketu příkazu pro operaci RMW (převzato z [4])

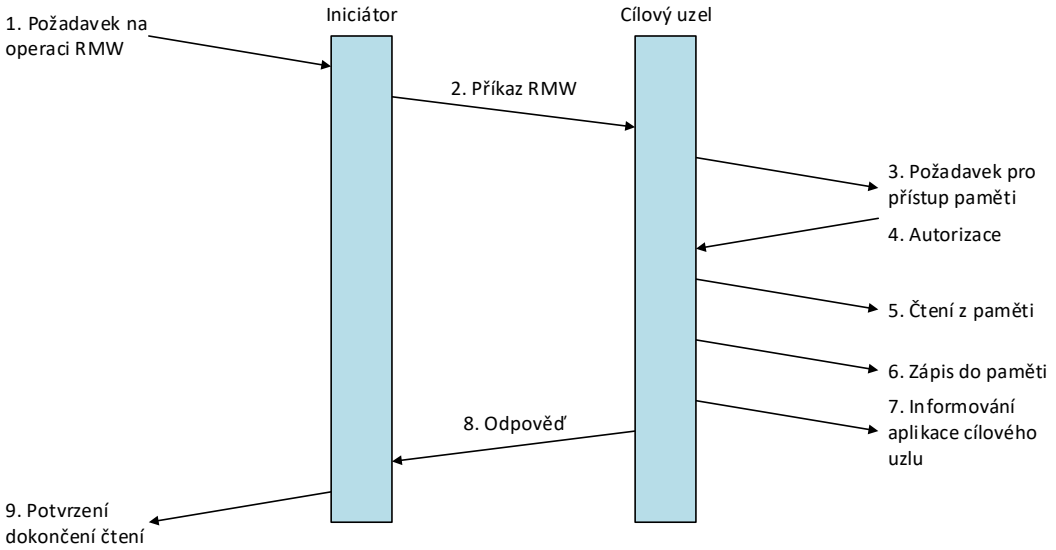
Na Obr. 2.10 lze vidět strukturu paketu odpovědi.

První odeslaný byte

	SpaceWire adresa iniciátoru	…	SpaceWire adresa iniciátoru
Logická adresa iniciátoru	Identifikátor protokolu	Instrukce	Status
Logická adresa cílového uzlu	Identifikátor transakce (MSB)	Identifikátor transakce (LSB)	Rezervováno = 0
Počet bytů dat (MSB) = 0	Počet bytů dat = 0	Počet bytů dat (LSB) = 0, 1, 2, 3, 4	CRC hlavičky
Data	Data	Data	Data
CRC dat	EOP		

Poslední odeslaný byte

Obr. 2.10: Struktura paketu odpovědi pro operaci RMW (převzato z [4])



Obr. 2.11: Sekvence událostí při provádění operace RMW (převzato z [4])

Stejně, jako v případě operace zápisu a operace čtení, může dojít k různým chybám, na které je třeba nějakým způsobem reagovat. Druhy chyb a chování iniciátoru a cílového uzlu při nich jsou shrnuty v Příloze C.

2.6 Chyby a stavové kódy

Během jednotlivých fází provádění požadované operace může dojít k určitým chybám. Příkladem můžou být chyby hlavičky příchozího paketu, jako například konec paketu uprostřed hlavičky nebo chyba hlavičky zjištěná kontrolou CRC hlavičky. Po detekci těchto chyb nelze s jistotou určit, jaký byl požadovaný druh operace nebo odkud paket přišel, a proto standard protokolu RMAP stanovuje, že při zjištění porušené hlavičky paketu analyzovaného cílovým uzlem má být tento paket zahozen a nemá být na něj odpovídáno (viz [4]).

V jiných případech je naopak žádoucí, aby byl iniciátor informován o dané chybě prostřednictvím odpovědi, která obsahuje stavový bajt, jenž lze k tomuto účelu využít. Odpověď je však odeslána pouze v tom případě, kdy si ji iniciátor vyžádá prostřednictvím zpracovaného příkazu (viz [4]).

Kromě odeslání odpovědi a zahození paketu jsou standardem jmenovány akce, jako přerušení nebo úplné zamezení přístupu do paměti, uložení informace o chybě v registrech cílového uzlu nebo informování jeho aplikace o chybě, pro více informací viz [4] nebo Přílohy A až C. Také je definováno chování iniciátoru při detekci chybné odpovědi. Standard definuje sadu stavových kódů, které mohou být obsaženy v poli stavu v paketu odpovědi. Tyto kódy jsou uvedeny v Tab. 2.4.

Tab. 2.4: Chybové a stavové kódy definované v protokolu RMAP (převzato z [4])

Kód	Typ stavu/chyby	Popis stavu/chyby	Typ Operace		
			Čtení	Zápis	RMW
0	Úspěch	Operace proběhla úspěšně.	x	x	x
1	Obecná chyba	Zjištěná chyba nespadá do žádného z ostatních typů nebo uzel nedokáže chybu přesněji určit.	x	x	x
2	Nedefinovaný typ RMAP paketu nebo příkazu	CRC hlavičky je v pořádku, ale typ paketu není ani příkaz ani odpověď nebo není daný příkaz protokolem RMAP používán.	x	x	x
3	Neplatný klíč	CRC hlavičky je v pořádku, ale přístupový klíč neodpovídá hodnotě očekávané aplikací cílového uzlu.	x	x	x
4	Chybné CRC dat	Chyba CRC bloku dat.	x		x
5	Brzké EOP	EOP znak zaznamenán před očekávaným koncem paketu.	x	x	x
6	Příliš dat	V příkazu se nachází více dat, než je očekáváno.	x	x	x
7	EEP	Detekován znak EEP, který říká, že při přenosu po síti došlo k chybě.	x	x	x
8	Rezervováno				
9	Zahlcení kontrolní vyrovnávací paměti	Byla požadována kontrola integrity dat, ale množství dat přesahuje kapacitu kontrolní vyrovnávací paměti.	x		x
10	RMAP příkaz neimplementován nebo neschválen	Aplikace cílového uzlu neschválila operaci. Důvodem může například být chybějící implementace daného příkazu.	x	x	x
11	Neplatná délka dat operace RMW	Množství dat RMW příkazu neodpovídá platným hodnotám 0, 2, 4, 6 nebo 8 bajtů.			x
12	Neplatná logická adresa cílového uzlu	CRC hlavičky je v pořádku, logická adresa cílového uzlu však neodpovídá žádné z adres, které daný uzel rozlišuje.	x	x	x
13 až 255	Rezervováno				

3 SBĚRNICE AMBA AHB

AMBA představuje soubor standardů sběrnicových rozhraní. Jeho tvůrcem je společnost ARM a obsahuje standardy pro různé druhy systémových či periferních sběrnic pro použití v rámci integrovaných obvodů.

Pro přístup realizovaných řadičů protokolu RMAP do paměti byla vybrána sběrnice AMBA AHB a řadiče DMA byly navrženy podle druhé revize standardu této sběrnice. Tato revize specifikuje úplné rozhraní AHB s přímou podporou více obvodů typu master. Novější revize specifikují odlehčené AHB-Lite rozhraní, u kterého lze realizovat tzv. multi-master systém nepřímými metodami, například tzv. vícevrstvým propojením.

3.1 Přehled

AHB protokol podporuje přenos dat v jednotlivých transakcích nebo v dávkách. Umožňuje volit bitovou šířku slova pro danou transakci. Dále podporuje prodloužení přenosu o čekací cykly z obou stran a přerušení přenosu kvůli chybě nebo dočasné neschopnosti obvodu slave přijmout či odeslat data, arbitraci přístupu jednotlivých obvodů master ke sběrnici včetně možnosti tzv. uzamčení přístupu pro danou transakci a poskytuje mechanismy pro ochranu dat a řízení přístupu k nim. Seznam signálů je uveden v Tab. 3.1.

Tab. 3.1: Seznam signálů AHB sběrnice (převzato z [5])

Název	Šířka [b]	Význam
HCLK	1	Sběrnicový hodinový signál
HRESETn	1	Reset
HADDR	32	Adresa
HTRANS	2	Typ přenosu
HWRITE	1	Zápis/Čtení
HSIZE	3	Šířka přenášeného slova
HBURST	3	Typ transakce
HPROT	4	Ochrana a přístup k datům
HREADY	1	Signály na sběrnici jsou platné
HRESP	2	Stav přenosu
HWDATA	W_{DAT}	Data pro zápis
HRDATA	W_{DAT}	Data pro čtení
HSEL	N_{SLV}	Výběr aktivního obvodu slave
HBUSREQ	N_{MST}	Požadavek o přístup ke sběrnici
HLOCK	N_{MST}	Požadavek o uzamčení transakce
HGRANT	N_{MST}	Požadavek o přístup přijat
HMASTER	4	Číslo obvodu master, který má současně přístup ke sběrnici
HMASTLOCK	1	Master provádí uzamčenou transakci
HSPLIT	16	Požadavek obvodu slave indikující, který master by měl získat přístup ke sběrnici pro obnovení přerušené transakce

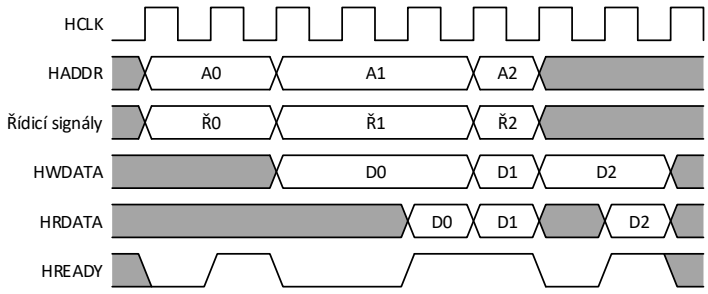
3.2 Standardní přenos

Přenos zahajuje master, následně je však řízen obvodem slave. Master udává platný přenos signálem HTRANS, směr toku dat signálem HWRITE, šířku přenášeného slova signálem HSIZE a délku transakce signálem HBURST. V Tab. 3.2 lze vidět hodnoty signálu HTRANS. Master nesmí nastavit šířku datového slova větší, než je šířka datové sběrnice. [5]

Tab. 3.2: Hodnoty signálu HTRANS (převzato z [5])

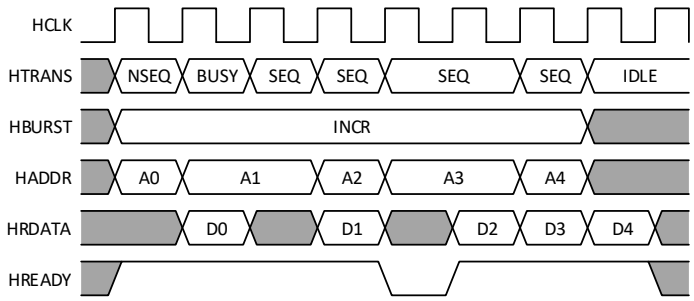
HTRANS[1]	HTRANS[0]	Název
0	0	IDLE
0	1	BUSY
1	0	NONSEQ
1	1	SEQ

Přenos je rozdělen na adresovou a datovou fázi. V adresové fázi je nastavena adresa a řídicí signály, v datové fázi jsou pak na sběrnici dostupná data. Přenosy jsou zřetězené tak, že během datové fáze jednoho přenosu probíhá adresová fáze následujícího přenosu. Veškeré signály jsou platné, když je aktivní signál HREADY. Slave může pomocí tohoto signálu implementovat čekací cykly. Master může implementovat čekací cykly v dávkové transakci nastavením signálu HTRANS na hodnotu BUSY. Příklad transakce s čekacími cykly lze vidět na Obr. 3.1. [5]



Obr. 3.1: Transakce s čekacími cykly (převzato z [5])

Pro transakce o jednom přenášeném slově nastaví master HTRANS na hodnotu NONSEQ a po ukončení přenosu jej nastaví zpět na hodnotu IDLE, pokud nenásleduje další transakce. Pro transakci v dávce master pro první přenos nastaví HTRANS na hodnotu NONSEQ a pokračuje se signálem HTRANS s hodnotou SEQ, která značí pokračování dávkové transakce. Příklad dávkové transakce znázorňuje Obr. 3.2. [5]



Obr. 3.2: Příklad dávkové transakce o nedefinované délce (převzato z [5])

Typ transakce a počet přenosů indikuje master nastavením signálu HBURST na příslušnou hodnotu. Seznam platných hodnot lze vidět v Tab. 3.3. Transakce SINGLE představuje transakci s jedním přenosem. Transakce INCR je dávková transakce, která nemá předem definovanou délku. INCR4 až INCR16 transakce jsou dávkovými transakcemi o pevně daném počtu přenosů. Totéž platí pro transakce typu WRAP. [5]

Tab. 3.3: Typy transakcí (převzato z [5])

HBURST	Název	Význam
0000_2	SINGLE	Transakce o jednom přenosu.
0001_2	INCR	Dávka o nedefinované délce s inkrementací adresy
0010_2	WRAP4	Dávka o 4 přenosech s přetečením adresy
0011_2	INCR4	Dávka o 4 přenosech s inkrementací adresy
0100_2	WRAP8	Dávka o 8 přenosech s přetečením adresy
0101_2	INCR8	Dávka o 8 přenosech s inkrementací adresy
0110_2	WRAP16	Dávka o 16 přenosech s přetečením adresy
0111_2	INCR16	Dávka o 16 přenosech s inkrementací adresy

Rozdíl mezi INCR a WRAP transakcemi je ten, že zatímco transakce INCR stále inkrementují adresu, pro adresu při transakcích WRAP platí, že adresa může přetéct na konci adresového prostoru, který odpovídá počtu přenášených bajtů v transakci a je na tento počet zarovnaný. Tab. 3.4 obsahuje příklad, který znázorňuje rozdíl mezi transakcemi typu INCR a WRAP o délce 8 přenosů a šířce slova 2 bajty. Pro zjednodušení má adresa pouze 8 bitů. [5]

Tab. 3.4: Příklad inkrementujicí adresy bez přetečení a s přetečením

Typ transakce	Adresa
INCR8	$C4_{16}$ $C6_{16}$ $C8_{16}$ CA_{16} CC_{16} CE_{16} $D0_{16}$ $D2_{16}$
WRAP8	$C4_{16}$ $C6_{16}$ $C8_{16}$ CA_{16} CC_{16} CE_{16} $C0_{16}$ $C2_{16}$

Na signál HTRANS s hodnotou IDLE je slave povinen odpovídat signálem HRESP o hodnotě OKAY. [5]

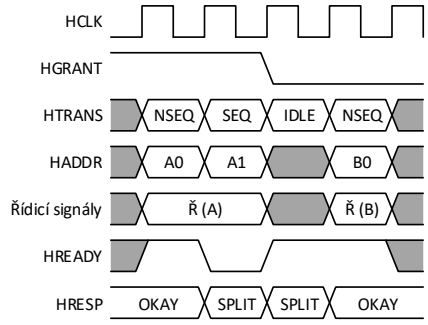
3.3 Přenos s přerušením

Během datové fáze přenosu poskytuje slave obvodu master stav přenosu prostřednictvím signálu HRESP. Obvykle má tento signál hodnotu OKAY, nicméně přenos může být přerušen z různých důvodů. Možné hodnoty signálu HRESP jsou v Tab. 3.5. [5]

Tab. 3.5: Hodnoty signálu HRESP (převzato z [5])

HRESP[1]	HRESP[0]	Název	Význam
0	0	OKAY	Přenos proběhl v pořádku.
0	1	ERROR	Při přenosu došlo k chybě.
1	0	RETRY	Přenos ještě neproběhl, master by se měl o přenos znovu pokusit.
1	1	SPLIT	Přenos je přerušen, master by měl znovu zažádat o přístup ke sběrnici a opakovat transakci od přerušeného přenosu.

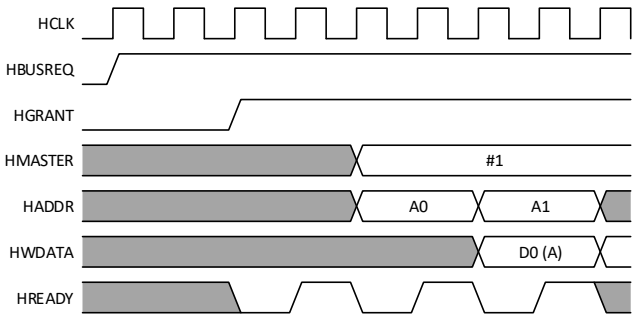
Pokud dojde k chybě při přenosu, signalizuje slave stav ERROR. Je-li potřeba, aby master znovu zahájil transakci, může slave signalizovat stav RETRY. V případě, že se slave rozhodne transakci z nějakého důvodu přerušit, signalizuje stav SPLIT. Ve všech třech případech je masteru odebrán přístup ke sběrnici. Kvůli zřetězení přenosů trvají tyto nestandardní odpovědi dva hodinové cykly, aby mohl master na tuto změnu zareagovat, viz Obr. 3.3. [5]



Obr. 3.3: Přerušení transakce (převzato z [5])

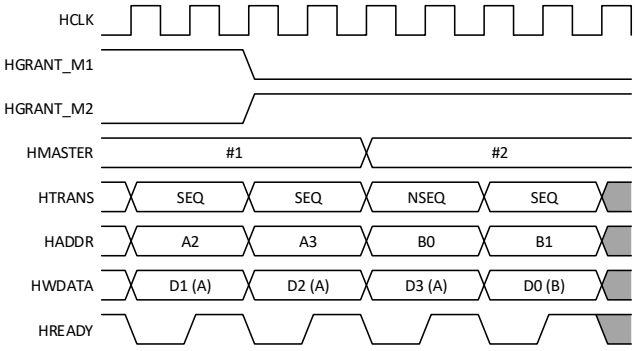
3.4 Arbitrace

Požaduje-li master přístup ke sběrnici, nastaví signál HBUSREQ, případně také HLOCK, na aktivní úroveň a čeká na odpověď arbitru. Arbitr přidělení přístupu signalizuje signálem HGRANT příslušícím danému obvodu master. Tento signál je platný, pokud je signál HREADY aktivní. Poté následuje adresová fáze prvního přenosu. Přidělení přístupu lze vidět na Obr. 3.4. [5]



Obr. 3.4: Přidělení přístupu ke sběrnici (převzato z [5])

Pokud master dále nepotřebuje přístup ke sběrnici, nastaví signál HBUSREQ na neaktivní úroveň. Pokud se arbitr rozhodne, že přidělí sběrnici jinému obvodu master, deaktivuje signál HGRANT. V následující fázi dojde k přepnutí adresových a řídicích signálů a o fázi později také k přepnutí datových signálů na sběrnici, viz Obr. 3.5. [5]



Obr. 3.5: Odebrání přístupu arbitrem (převzato z [5])

3.5 Endianita

Endianita udává způsob řazení bajtů dat v paměti. Základní členění rozlišuje tzv. Big Endian, kde bajt nejvyššího významu (MSB) je uložen na nejnižší adrese v paměti, a tzv. Little Endian, kde bajt nejnižšího významu (LSB) je uložen na nejnižší adrese v paměti. Tento způsob není nijak standardizován a každý systém je navržen různě pro určitý způsob řazení bajtů. Například procesory s architekturou Intel x86 používají řazení typu Little Endian, zatímco architektura PowerPC řadí data v paměti ve formátu Big endian. Existují také architektury, které podporují oba druhy endianity, příkladem je Cortex-M0 od již zmíněné společnosti ARM. Výsledný způsob řazení dat v paměti pak záleží na příslušné implementaci procesoru. [6] [7] [8]

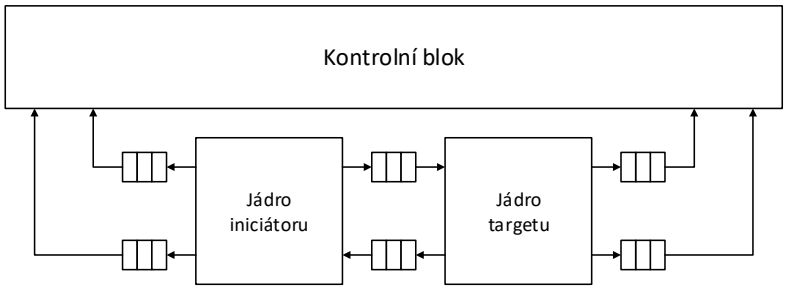
Standard protokolu AHB specifikuje endianitu typu Little Endian (LE) a Big Endian o šířce slova 32 bitů (BE-32), viz [5]. Řazení bajtů dat na datové sběrnici pro tyto typy endianity je znázorněno v Tab. 3.6.

Tab. 3.6: Řazení bajtů na datové sběrnici pro různé typy endianity, viz [5]

Šířka slova [b]	**Adresový offset a endianita systému**							
	LE				**BE-32**			
	3	**2**	**1**	**0**	**0**	**1**	**2**	**3**
32	B3	B2	B1	B0	B3	B2	B1	B0
16			B1	B0	B1	B0		
	B1	B0					B1	B0
8				B0	B0			
			B0			B0		
		B0					B0	
	B0							B0

4 NÁVRH VERIFIKAČNÍCH MODELŮ

Pro efektivní testování navržených řadičů RMAP iniciátoru a cílového uzlu bylo třeba navrhnout verifikační modely těchto řadičů. Tyto modely byly navrženy a vytvořeny v jazyce SystemVerilog, který je v dnešní době široce využíván pro funkční verifikaci.



Obr. 4.1: Topologie pro ověření verifikačních modelů iniciátoru a cílového uzlu

Na Obr. 4.1 lze vidět topologii pro ověření funkce verifikačních modelů. Pro snazší synchronizaci při předávání dat je v rámci modelů použita kolekce schránky, tzv. mailbox, která slouží pro komunikaci mezi procesy. To umožňuje jednotlivým komponentám běžet souběžně a data předávat nebo přijímat podle potřeby. Schránky jsou použity pro následující rozhraní:

- Rozhraní mezi modely iniciátoru a cílového uzlu slouží pro posílání paketů příkazu a odpovědi. Tyto pakety jsou reprezentovány datovou strukturou obsahující pole s přenášenými bajty paketu a dva bity signalizující druh konce paketu.
- Rozhraní mezi modelem iniciátoru a kontrolním blokem slouží k předání informací o odeslaném paketu příkazu cílovému uzlu a o přijatém a zpracovaném paketu odpovědi, odeslaného cílovým uzlem, včetně zjištěných chyb.
- Rozhraní mezi modelem cílového uzlu a kontrolním blokem slouží k předání informací o odeslaném paketu odpovědi iniciátoru a o přijatém a zpracovaném paketu příkazu, odeslaného iniciátorem, včetně zjištěných chyb.

4.1 Iniciátor

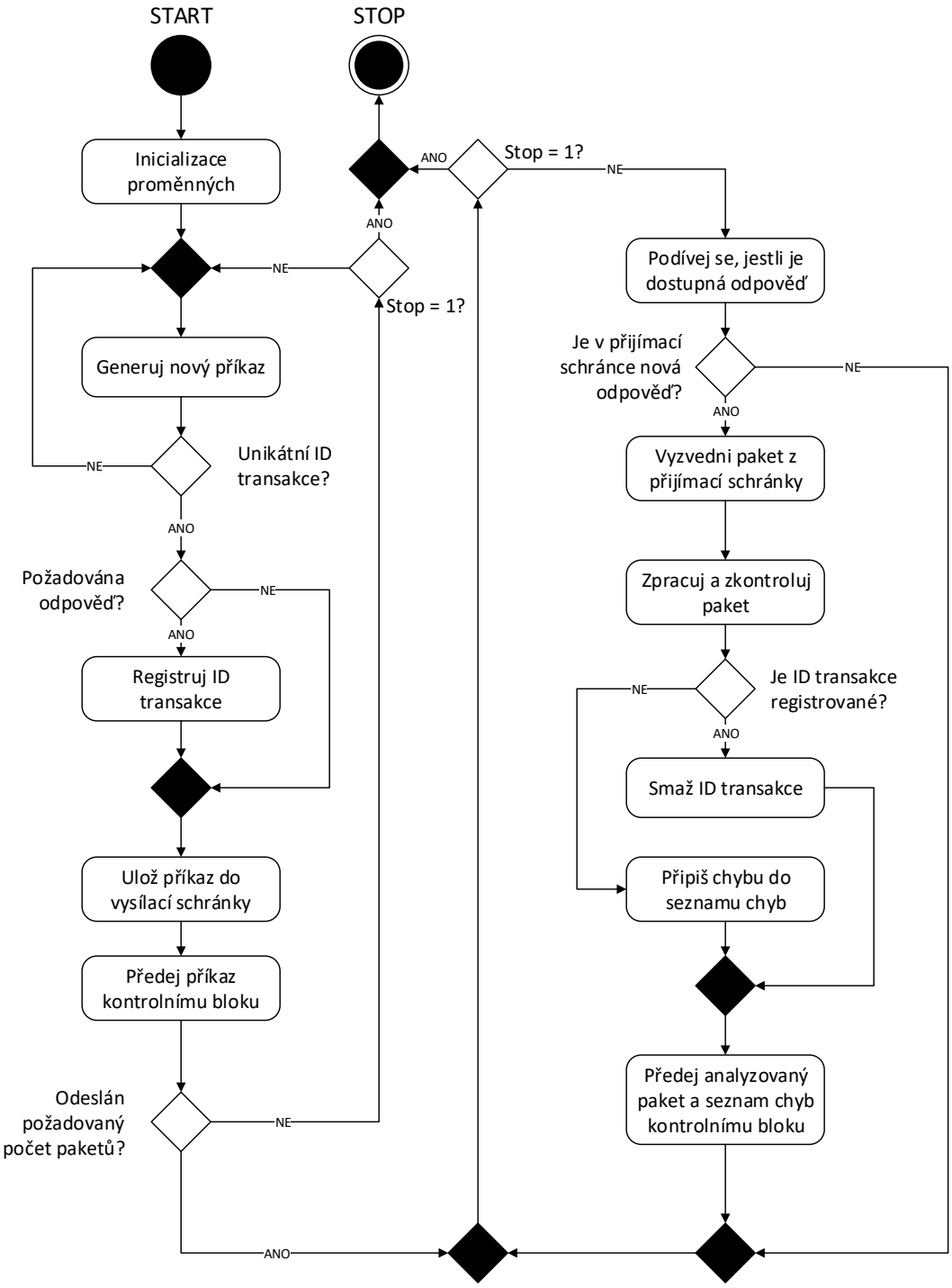
Model iniciátoru má za úkol generovat pakety příkazů a ty odesílat cílovému uzlu. Poté čeká, zda z cílového uzlu přišel paket odpovědi a ten analyzuje. Během analýzy ukládá jednotlivé bajty paketu do příslušné datové struktury a zároveň registruje případné chyby. Analyzovaný paket a kolekci nalezených chyb pak vrací hlavní rutině iniciátoru ke zpracování, která paket ve formě datové struktury a seznam zjištěných chyb paketu odešle monitoru.

Pakety příkazů jsou generovány pomocí randomizace, jsou tedy generovány náhodně, přičemž proces generace je řízen určitými omezeními. Těmito omezeními jsou například zákaz používání adresy pro SpaceWire, maximální počet bajtů dat, maximální platná adresa apod. Iniciátor umožňuje omezit generaci pouze na platné pakety, anebo povolit generaci paketů s chybami. Testovací prostředí při spuštění iniciátoru specifikuje mimo jiné také počet příkazů k odeslání.

Po spuštění se iniciátor chová následujícím způsobem. Nejprve inicializuje všechny proměnné použité k řízení generace příkazů. Poté na základě těchto proměnných vygeneruje příkaz a převede ho na posloupnost bajtů. Přitom zkontroluje, jestli vygenerovaný identifikátor transakce již není registrován a pokud ano, proces generace je zopakován. Pokud odeslaný příkaz požaduje od cílového uzlu odpověď, je identifikátor transakce registrován. Poté je vytvořená posloupnost bajtů uložena do schránky vysílače a náhodně vygenerovaný popis paketu je uložen do schránky monitoru. Tento proces se opakuje, dokud není odeslán počet bajtů specifikovaný testovacím prostředím při spuštění iniciátoru.

Po odeslání všech paketů přejde iniciátor do přijímací fáze, kdy čeká, dokud není ve schránce přijímače dostupný paket odpovědi. Tuto odpověď si pak převezme a započne s její analýzou a zpracováním. Při detekci chyby tuto chybu zaregistruje a zachová se podle úkonů definovaných v Přílohách A až C. Po dokončení analýzy odešle zpracovanou odpověď ve formě struktury a kolekce chyb monitoru prostřednictvím schránky.

Provádění procedury může testovací prostředí zastavit pomocí funkce, která nastaví příznakový bit pro zastavení iniciátoru. Jakmile hlavní rutina iniciátoru zjistí, že tento příznakový bit je platný, ukončí svoji činnost. Chování hlavní rutiny iniciátoru je znázorněno v diagramu na Obr. 4.2.



Obr. 4.2: Diagram hlavní rutiny modelu iniciátoru

4.2 Cílový uzel

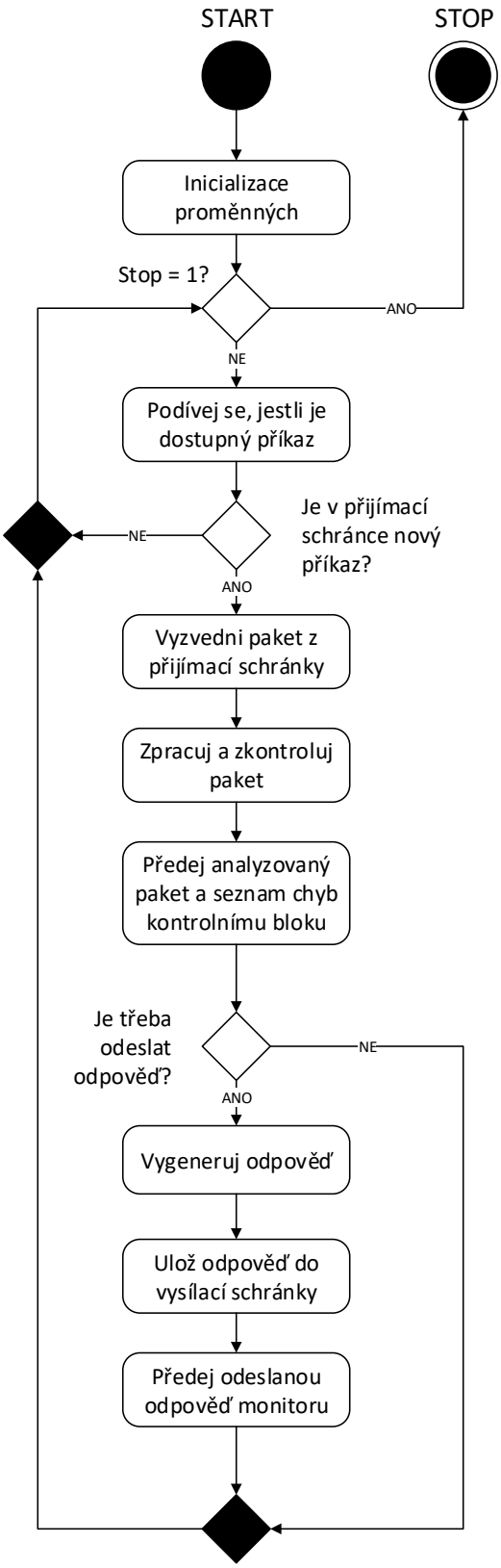
Úkolem modelu cílového uzlu je přijímat příkazy od iniciátoru, zkontrolovat je, tyto příkazy provést nebo odmítnout a požaduje-li to iniciátor, poslat zpět odpověď s výsledkem operace.

Stejně jako v případě iniciátoru jsou pakety odpovědi generovány pomocí procesu randomizace, nicméně jeho omezení jsou daleko striktnější, jelikož odpověď vychází ze zpracovaného příkazu. Náhodného generování je využito zejména pro generaci odpovědí s chybami.

Po spuštění cílový uzel inicializuje proměnné omezující proces generace odpovědi, a poté vstoupí do cyklu, jenž lze ukončit stejným způsobem, jako v případě iniciátoru, tj. nastavením příznakového bitu pro ukončení běhu modelu. V tomto cyklu cílový uzel nejdříve kontroluje, zda je ve schránce přijímače dostupný paket příkazu od iniciátoru. Pokud ano, tento paket, stejně jako iniciátor, analyzuje a zpracuje, přičemž registruje případné chyby a reaguje na ně způsobem uvedeným v Přílohách A až C. Pokud je paket v pořádku a aplikace cílového uzlu schválila přístup do paměti, je provedena požadovaná operace. Analyzovaný paket a soubor detekovaných chyb jsou pak předány monitoru skrze jeho schránku.

Poté cílový uzel na základě analyzovaného příkazu vygeneruje odpověď v případě, že je to iniciátorem požadováno a nebyla-li detekována chyba hlavičky příkazu. V takovém případě totiž nelze spolehlivě určit typ příkazu ani jeho iniciátor. Vygenerovanou odpověď pak uloží do schránek vysílače a monitoru. Chování hlavní rutiny modelu cílového uzlu je znázorněno v diagramu na Obr. 4.3.

Kromě kontroly chyb provádí model cílového uzlu také autorizaci daného příkazu. Kontroluje, zda byla specifikována správná logická adresa a správný přístupový klíč. Další způsoby autorizace pak standard převádí na příslušnou aplikaci cílového uzlu, stejně tak obsah stavového slova v odpovědi v případě detekce více druhů chyb je aplikačně specifický. Aplikace také může implementovat různý způsob modifikace dat při RMW operaci nebo se chovat odlišně při přístupu do paměti. Proto byla definována třída aplikace cílového uzlu, která obsahuje funkce, jež jádro modelu používá. Tyto funkce jsou definovány jako virtuální, aby uživatel modelu mohl pomocí dědičnosti z bázové třídy aplikace odvodit svoji vlastní třídu se svojí implementací těchto funkcí. Jádro modelu ve svém konstruktoru pak vyžaduje referenci na objekt typu bázové třídy aplikace, jejíž funkce volá. Pomocí polymorfismu tak lze měnit chování aplikace cílového uzlu podle potřeby.



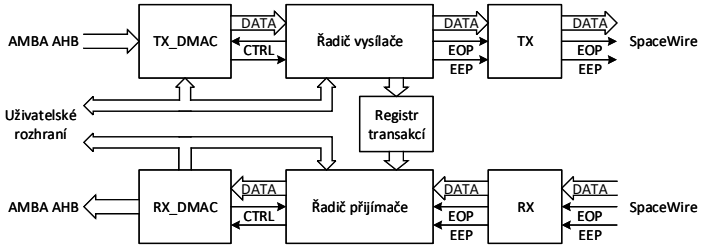
Obr. 4.3: Diagram hlavní rutiny modelu cílového uzlu

5 NÁVRH ŘADIČŮ PROTOKOLU RMAP

Hlavním cílem této práce je návrh a realizace řadičů protokolu RMAP. Oba řadiče jsou navrženy jako synchronní obvody s jedním hodinovým signálem a u některých schémat tak nejsou v této práci pro lepší přehlednost těchto schémat signály hodin a resetu znázorněny. Řadiče jsou z hlediska toku dat navrženy tak, že předávání dat mezi jednotlivými částmi probíhá metodou v odborné terminologii nazývanou handshake.

5.1 Návrh řadiče iniciátoru

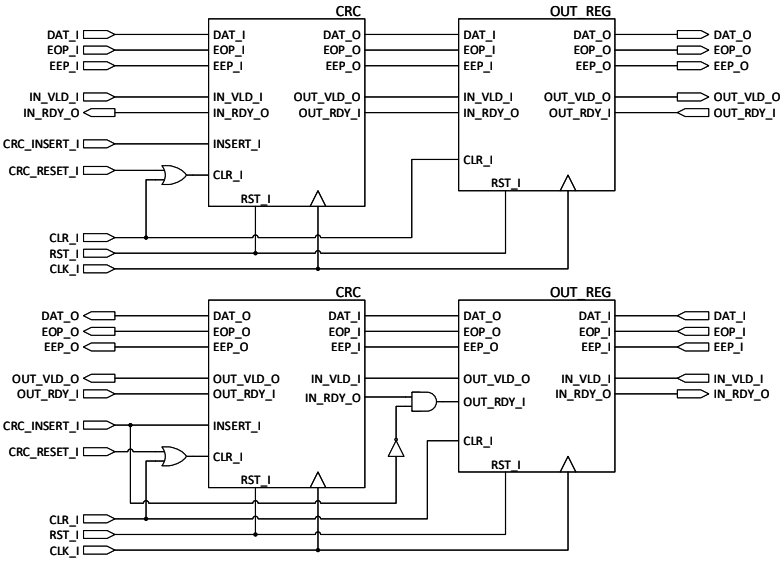
Řadič iniciátoru protokolu RMAP je rozvržen do několika základních bloků. Blokové schéma architektury iniciátoru znázorňuje Obr. 5.1. Jak lze vidět, architektura je primárně rozčleněna na vysílací a přijímací směr. Na pravé straně se nachází rozhraní pro navazující řadič sítě SpaceWire, na levé pak sběrnicové a uživatelské rozhraní.



Obr. 5.1: Architektura iniciátoru protokolu RMAP

5.1.1 Bloky TX a RX

Tento blok představuje přenosovou část vysílače. Obsahuje zejména jednotku pro výpočet CRC a má řetězovou strukturu o dvou stupních. Prvním je již zmíněná jednotka CRC, druhý stupeň dělí datovou cestu mezi modulem SpaceWire a navrženým řadičem pomocí registrů. Strukturu bloků RX a TX znázorňuje Obr. 5.2.



Obr. 5.2: Struktura bloků TX (nahoře) a RX (dole)

5.1.2 CRC

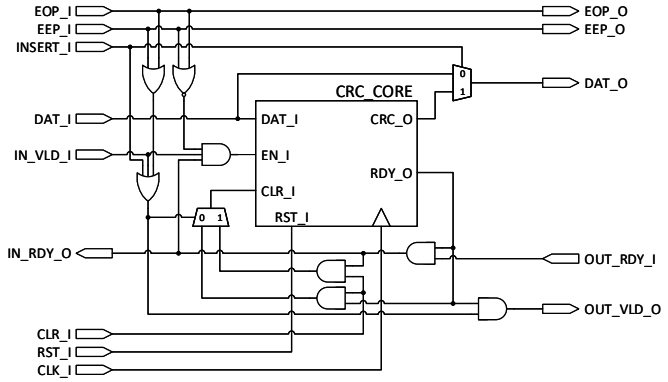
Jednotka CRC je součástí bloků RX a TX a byla navržena tak, aby ji bylo možno snadno řídit blokem připojeným na jejím vstupu, v případě vysílače řadičem TX_CTRL. Jednotka obsahuje vstup pro data, znaky EOP a EEP a dále vstupy pro nulování jednotky a vložení vypočítaného CRC do řetězce. Schéma jednotky CRC lze vidět na Obr. 5.3.

Jednotka pracuje následujícím způsobem. Při odesílání dat je pro indikaci platnosti použit vstup IN_VLD_I. Je-li jednotka CRC připravena tato data přijmout, nastaví signál IN_RDY_O do log. 1 a v případě, že signál IN_VLD_I signalizuje platná data, dojde v následujícím taktu hodinového signálu k zachycení datového slova pro přepočítání kontrolního součtu a zároveň k přenosu vstupního datového slova přímo do následujícího stupně. Toho je dosaženo tím, že signál IN_RDY_O je závislý nejen na stavu výpočetního jádra CRC, ale také na stavu následujícího stupně, který je indikován signálem OUT_RDY_I.

Má-li být přenesen znak EOP nebo EEP, tento znak blokuje výpočet kontrolního součtu, jelikož vstupní data nejsou platná.

V případě, že je potřeba zařadit vypočítaný kontrolní součet do paketu, nastaví řídící blok vstup INSERT_I na hodnotu log. 1. Vypočítaný kontrolní součet je na výstup přenesen jeden hodinový takt poté, co jsou signály IN_RDY_O a INSERT_I ve shodě. V tomto stavu je datový výstup výpočetního jádra jednotky CRC přiveden na datový výstup pomocí multiplexoru.

Nulování jednotky CRC probíhá stejně jako v případě odeslání vypočteného kontrolního součtu, místo signálu INSERT_I je však nastaven signál CLR_I.

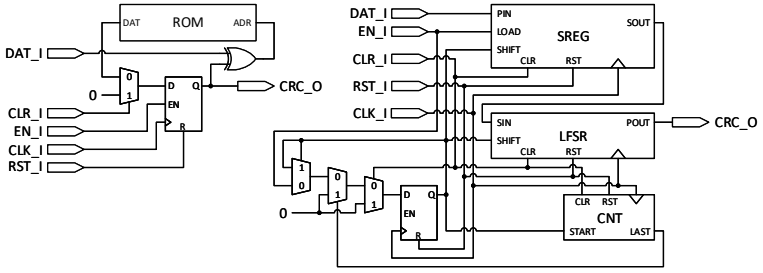


Obr. 5.3: Jednotka CRC

Výpočetní jádro bylo navrženo pro dvě možné implementace. První variantou je implementace pomocí registru LFSR, která byla popsána v kapitole 2.1.12. Druhou variantou je paralelní implementace využívající náhledové tabulky zapojené ve zpětné vazbě, jejíž adresa je výsledkem operace nonekvivalence mezi vstupním datovým slovem a současným mezivýsledkem kontrolního součtu uloženým v registru. Obě implementace lze vidět na Obr. 5.4.

Ze schématu na Obr. 5.3 vyplývá, že rozhraní jednotky CRC je navrženo pro řízení blokem na vstupu, což ale znamená, že nelze tak snadno tuto jednotku řídit v případě přijímače iniciátoru, jelikož jeho řadič se nachází na výstupu jednotky CRC. To je řešeno pomocí manipulace rozhraní na vstupu jednotky CRC skrze hradlo AND řízené signálem INSERT tak, aby blokovalo signál RDY na vstupu jednotky CRC a zabránilo tak přenosu dat z jejího vstupu, čímž by v případě požadavku na kontrolní součet došlo ke ztrátě dat. Toto blokování je znázorněno na Obr. 5.2.

Při potřebě získání kontrolního součtu tak lze nastavit signál INSERT tak, jako obvykle, a v případě nulování je potřeba krom signálu CLR nastavit i signál INSERT, aby došlo k výše zmíněnému blokování.



Obr. 5.4: Implementace výpočtu CRC – paralelní (vlevo) a LFSR (vpravo) varianta

5.1.3 Řadič vysílače

Jedná se o řadič paketů protokolu RMAP. Zprostředkovává uživatelské aplikaci generaci paketu příkazu a jeho odeslání cílovému uzlu. Provádí kontrolu vstupních dat uživatelské aplikace, z těchto dat a za pomoci řadiče DMA vysílače vytváří příslušné pakety příkazů a je-li požadována odpověď, ukládá relevantní data o příkazu k pozdějšímu zpracování při přijetí odpovědi. Jeho schéma lze vidět v Příloze D.

Řadič obsahuje bloky pro kontrolu vstupních dat uživatelské aplikace, jakými jsou např. délka adresy odpovědi v síti SpaceWire, formát instrukčního slova specifikující typ operace v paketu, délka dat pro operaci RMW nebo kontrola vstupních dat pro řadič DMA. Dále obsahuje čítače pro odpočet odeslaných bajtů v jednotlivých částech paketu příkazu, přičemž některé tyto čítače disponují funkcemi specifickými pro danou část paketu, jako je např. výstup pro výběr bajtu masky pro RMW operaci. Další částí jsou posuvný registr načítající bajty hlavičky z uživatelské aplikace a odesílající je v daném pořadí dále a výstupní registry poskytující data a řídicí signály navazujícímu stupni, v tomto případě bloku TX. Hlavní jednotkou tohoto řadiče je stavový automat, který řídí celý jeho chod.

Při požadavku řadič nejprve kontroluje, zda není v poskytnutých datech žádná chyba. Pokud ano, oznámí uživatelské aplikaci konec činnosti včetně výsledku kontroly chyb a přejde opět do stavu nečinnosti. Jsou-li vstupní data v pořádku a jedná se o operaci s požadavkem na odpověď, zapíše nejprve relevantní data do registru transakcí. Pokud není požadována odpověď, přeskočí tento krok a rovnou načte první část hlavičky do posuvného registru. Pokud je součástí paketu příkazu adresa sítě SpaceWire, načte ji

pomocí řadiče DMA z paměti, odešle ji a až poté pokračuje k odeslání první části hlavičky uložené v posuvném registru. Po odeslání první části hlavičky načte z paměti adresu sítě SpaceWire pro paket odpovědi a odešle ji, je-li aplikací požadována. Následuje postupné načtení a odeslání zbylých tří částí hlavičky s následným vřazením kontrolního součtu z jednotky CRC a posléze jejím nulováním.

Po odeslání hlavičky záleží na typu operace. Jedná-li se o operaci zápisu, načte z paměti data a odešle je, v případě operace RMW navíc znovu inicializuje datový čítač a odešle masku. Následně připojí kontrolní součet, vynuluje jednotku CRC, a nakonec odešle znak EOP značící konec paketu. V případě operace čtení uvedené kroky přeskočí a odešle znak EOP. Následně řadič informuje uživatelskou aplikaci o dokončení operace a přejde do stavu nečinnosti.

Během přijímání dat z řadiče DMA může dojít k chybě při přístupu do paměti. V takovém případě řadič okamžitě přeruší odesílání paketu, odešle znak EEP, vynuluje jednotku CRC a indikuje konec operace s následným přechodem do stavu nečinnosti. Chyba sběrnice je v řadiči uložena do registru a z něj poskytnuta uživatelské aplikaci.

Řadič umožňuje vymazat údaje o odeslaném příkazu z registru transakcí na základě poskytnutého identifikátoru transakce. V takovém případě řadič vymaže daný příkaz z registru, je-li tam opravdu uložen, a poté indikuje konec činnosti s následným přechodem do stavu nečinnosti.

5.1.4 Řadič přijímače

Podobně jako řadič vysílače, pracuje řadič přijímače se samotnými pakety. Jeho účelem je příjem paketů odpovědi, jejich kontrola a zpracování. Schéma zapojení tohoto bloku je znázorněno v Příloze E.

Blok obsahuje blok REG, který slouží k uchování údajů hlavičky přijímaného paketu a výsledků ověření obou kontrolních součtů paketu. Krom toho kontroluje, zda je typ operace v platném formátu, zda výsledky výpočetní jednotky CRC mají po sečtení všech bajtů a příslušného kontrolního součtu v paketu nulovou hodnotu a zda má přijatý paket odpovídající identifikátor protokolu. Také porovnává typ operace a logickou adresu cílového uzlu v odpovědi s hodnotami uloženými v registru transakcí na základě přijatého identifikátoru transakce.

Další částí řadiče přijímače je čítač počtu přijatých datových bajtů. Tento čítač je inicializován počtem datových bajtů, který je součástí hlavičky přijatého paketu, a postupně odpočítává přijímané bajty dat. Čítač má dva výstupy. Jeden hlásí, že následující bajt je dle údajů v hlavičce poslední, druhý udává, že jsou v rámci paketu očekávána data. Čítač dekrementuje při přenosu datového bajtu do výstupního registru, kde je připraven pro odebrání řadičem DMA.

K tomuto rozhraní se také váže registr, který prostřednictvím portu VLD_O signalizuje platnost dat v datovém registru, a registr pro indikaci posledního bajtu dat. Tato indikace je realizována pomocí portu LAST_O a pro zajištění délky aktivní úrovně

pro indikaci posledního bajtu je zmíněný registr doplněn detektorem hrany jakožto tvarovacím obvodem. Signál LAST_O je pak aktivní, dokud řadič DMA nepřevezme poslední datový bajt. Výstupní registry jsou pak aktualizovány pouze, je-li signál VLD_O neaktivní, anebo v případě, kdy je řadič DMA připraven přijmout nová data. Od této podmínky se pak také odvíjí způsob předávání bajtů dat mezi jednotkou CRC a řadičem přijímače.

Poslední významnou částí řadiče přijímače je jeho řídicí blok. Řadič přijímače nejprve ukládá jednotlivé bajty hlavičky paketu do registrů v bloku REG. Poté načte z jednotky CRC výsledný kontrolní součet, který krom bajtů hlavičky zahrnuje i kontrolní součet této hlavičky obsažený v paketu. Po uložení výsledku nuluje jednotku CRC a zároveň kontroluje chyby hlavičky. Pokud je v hlavičce paketu chyba nebo se jedná o odpověď na příkaz pro operaci zápisu, počká na zakončovací znak, nastaví požadavek pro uživatelskou aplikaci na převzetí údajů o přijatém paketu a vyčká na reakci uživatelské aplikace. Pokud jako zakončovací znak přijme znak EEP, dodatečně nastaví výstup pro indikaci chybového konce paketu.

Pokud přijatý paket přísluší operaci čtení nebo RMW, řadič přijímače spustí řadič DMA a zahájí přenos dat. Přitom používá datového čítače a jeho indikátoru posledního bajtu jako reference pro určení konce paketu a chyby v případě, kdy je paket příliš krátký nebo dlouhý. Po přenosu dat řadič čeká, dokud řadič DMA nedokončí přenos dat do paměti. Následuje sekvence ověření kontrolního součtu dat. Je kontrolován konec paketu a je-li paket příliš dlouhý, tj. na vstupu se nenachází ani jeden z ukončovacích znaků, zahodí automat zbylá příchozí data s tím, že počká na zakončovací znak. Poté indikuje špatnou délku dat, případně chybový konec paketu při přijetí znaku EEP.

V případě, že je kontrolní součet hlavičky v pořádku a identifikátor protokolu neodpovídá identifikátoru protokolu RMAP, je celý paket zahozen bez upozornění uživatelské aplikace. Toto je také jediná situace, kdy řadič přijímače nečeká na potvrzení přijetí informací o paketu uživatelskou aplikací. Je-li během zpracování hlavičky přijat zakončovací znak, přejde automat do stavu, ve kterém indikuje předčasný konec paketu, případně chybový konec paketu při přijetí znaku EEP. Dojde-li k chybě při přístupu do paměti, řadič DMA tuto skutečnost indikuje řadiči přijímače, který následně zahodí zbytek paketu a oznámí chybu uživatelské aplikaci. Chyba přístupu do paměti je přitom uložena do vyhrazeného registru, dokud uživatelská aplikace nepotvrdí přijetí informací o paketu.

5.1.5 Registr transakcí

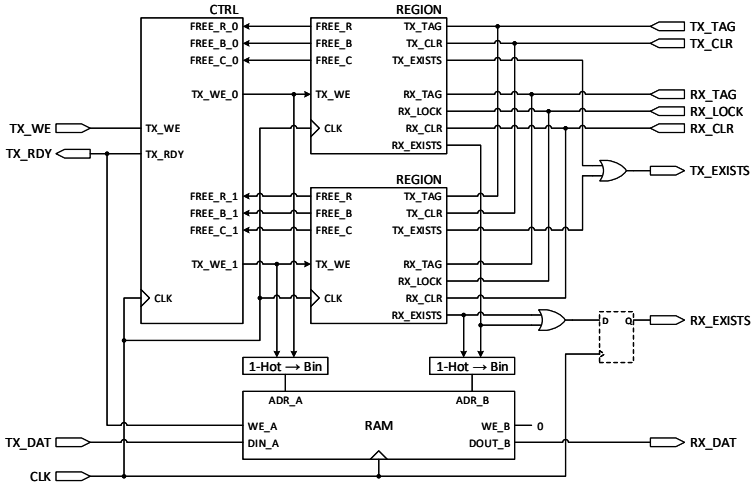
Registr transakcí je blok, který slouží k uchování důležitých údajů o odeslaném paketu příkazu, a to proto, aby mohl přijímač při přijetí paketu odpovědi tuto odpověď spárovat s příslušným příkazem. Data jsou do něj tedy ukládána pouze v případech, kdy odeslaný příkaz požaduje odpověď.

Dle [3] má paket odpovědi totožný identifikátor transakce, jako paket příkazu, který tuto odpověď požadoval. Proto je potřeba k údajům přistupovat na základě

tohoto identifikátoru. Jednoduché ukládání identifikátoru a příslušných údajů do stejné paměti vyžaduje sekvenční prohledávání této paměti, které v závislosti na její kapacitě může být příliš dlouhé. Opačným extrémem je použití identifikátoru jako adresy paměti, jelikož identifikátor transakce má šířku 16 bitů a je pro větší flexibilitu poskytován uživatelskou aplikací. Takové řešení by vyžadovalo paměť o poměrně velké kapacitě, která nemusí být v mnoha aplikacích zdaleka využita.

Z těchto důvodů byl registr transakcí navržen jako asociativní paměť, ke které lze přímo přistupovat pomocí identifikátoru transakce a která může mít volitelnou kapacitu. Kapacita registru pak pouze omezuje počet transakcí, které požadují odpověď.

Blokové schéma registru transakcí znázorňuje Obr. 5.5. Základem je dvouportová paměť RAM, která může být v případě cílového FPGA realizována pomocí LUT, anebo může využít dostupných paměťových bloků. V takovém případě na výstupu RX_EXISTS přibude jeden registr kvůli synchronizaci tohoto signálu se čtenými daty. Tato paměť má jeden port vyhrazený pro zápis a druhý pro čtení. K zápisovému portu přistupuje vysílač, ke čtecímu portu pak přijímač.

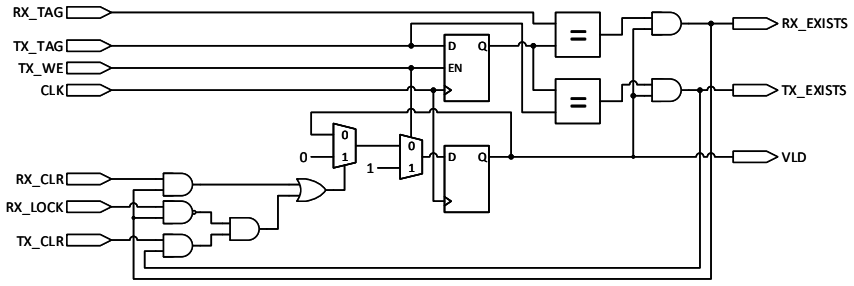


Obr. 5.5: Registr transakcí

Další významnou částí je kontrolér, který řídí zápis údajů o transakci ze strany vysílače. Zápis probíhá tak, že vysílač požádá o registraci příkazu s daným identifikátorem. Pokud již existuje platný záznam s daným identifikátorem, kontrolér je nečinný. Signál udávající existenci záznamu s daným identifikátorem je poskytnut vysílači, díky čemuž vysílač může přerušit svoji činnost a ohlásit chybu uživatelské aplikaci. Pokud není žádný příkaz s daným identifikátorem zaregistrován, kontrolér začne prohledávat registr, zda není některá z buněk volná. Pro urychlení prohledávání jsou buňky rozděleny do tzv. bloků a ty pak do tzv. regionů, viz Obr. 5.7. Každá z těchto entit poskytuje na své úrovni signál o tom, zda je v dané skupině buněk nebo bloků volné místo. Zrychlení hledání volného místa v registru transakcí pak spočívá v tom, že kontrolér nejprve prohledává registr na úrovni regionů. Pokud je v některém z těchto regionů volné místo, začne hledat na úrovni bloků v tomto regionu a poté

na úrovni buněk. Pomocí dekodéru pak generuje signál pro povolení zápisu do dané buňky, přičemž signalizuje vysílači, že dojde k zápisu identifikátoru transakce do buňky. K samotnému zápisu pak dojde o jeden takt hodinového signálu později.

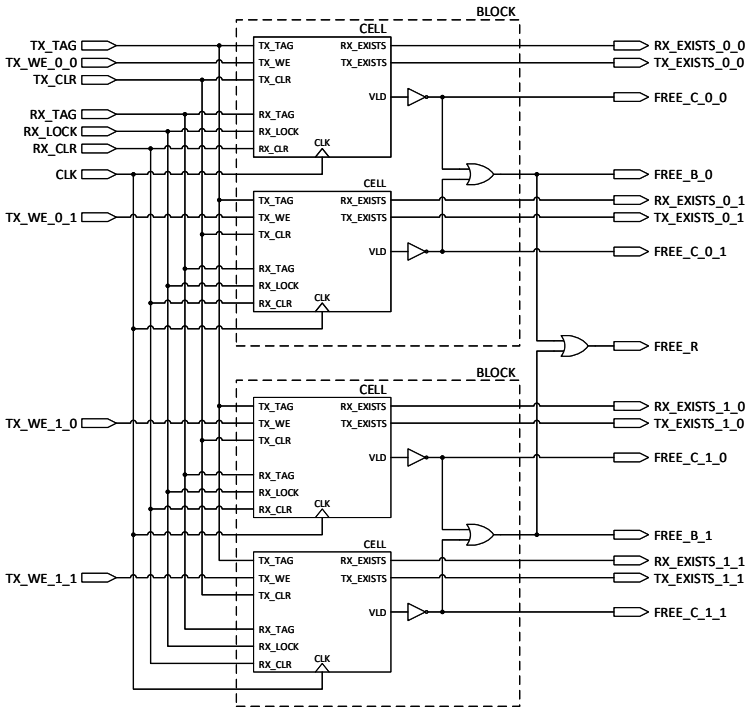
Na Obr. 5.6 lze vidět strukturu buňky uchovávající identifikátor transakce. Skládá se zejména z 16bitového registru pro uchování identifikátoru a 1bitového registru, který udává, zda je daný záznam platný. Pro vyhledávání slouží dva komparátory, každý pro jednu přistupující stranu, aby bloky vysílače i přijímače mohly vyhledávat nezávisle na sobě. Signály RX_EXISTS a TX_EXISTS signalizují shodu s daným identifikátorem za předpokladu, že daný záznam je platný. Při registraci transakce je nastaven signál TX_WE na hodnotu log. 1, což způsobí, že se do registru TAG uloží identifikátor, do registru VLD se zapíše log. 1 a záznam je tak uložen a označen za platný. Záznam lze vymazat za předpokladu, že identifikátor odpovídá obsahu registru TAG, přičemž je záznam platný. Vymazat lze záznam z obou stran, ale pro zabránění smazání záznamu uživatelskou aplikací v době zpracovávání odpovědi se stejným identifikátorem je buňka opatřena zamykacím mechanismem, kterým může přijímač zabránit smazání ze strany vysílače.



Obr. 5.6: Registr transakcí – paměťová buňka

Strukturu členění buněk do bloků a těch do jednotlivých regionů znázorňuje Obr. 5.7. Vstupy pro identifikátor, mazání záznamu a jeho blokování jsou společné pro všechny buňky v registru, jelikož ten je navržen tak, aby v jednom momentě měl každý platný záznam unikátní identifikátor a tyto signály mají účinek pouze pro buňku, která obsahuje daný identifikátor a záznam v ní je platný. Každá buňka má pak svůj signál pro povolení zápisu.

Na Obr. 5.7 lze také vidět způsob, jakým dané bloky a regiony indikují volné místo. Na úrovni bloku jsou využity signály VLD z jednotlivých buněk, které jsou invertovány a indikují tak volné buňky. Tyto signály jsou pak vyvedeny pro každou buňku zvlášť pro vyhledávání volného místa na úrovni buněk a také sloučeny pomocí logického součtu do jednoho signálu reprezentujícího volné místo v daném bloku. Na úrovni regionu je situace obdobná, jen místo buněk jsou seskupovány bloky. Výstupem regionu je pak množina signálů indikujících volné místo v regionu, jednotlivých blocích, a nakonec v jejich buňkách, a dále také signály EXISTS, které signalizují shodu poskytnutého identifikátoru s identifikátorem uloženým v příslušné buňce.



Obr. 5.7: Registr transakcí – členění buněk v rámci regionu

Pro adresování paměti je na straně vysílače použito signálů pro povolení zápisu vedoucích z kontroléru, na straně přijímače pak signálů EXISTS. Oba soubory signálů jsou převedeny na adresu převodníkem z kódu 1 z N na binární kódování. To znamená, že registr transakcí umožňuje uložit až N záznamů, přičemž pro *N* platí

$$N_{REC} = N_R \cdot N_B \cdot N_C, \tag{5.1}$$

kde N je celkový počet buněk, N_R je počet regionů, N_B počet bloků v regionu a N_C je počet buněk v jednom bloku. Kapacita paměti je určen podle vztahu

$$2^{W_{ADR}} = 2^{\mathrm{ceil}(\log_2 N_{REC})}, \tag{5.2}$$

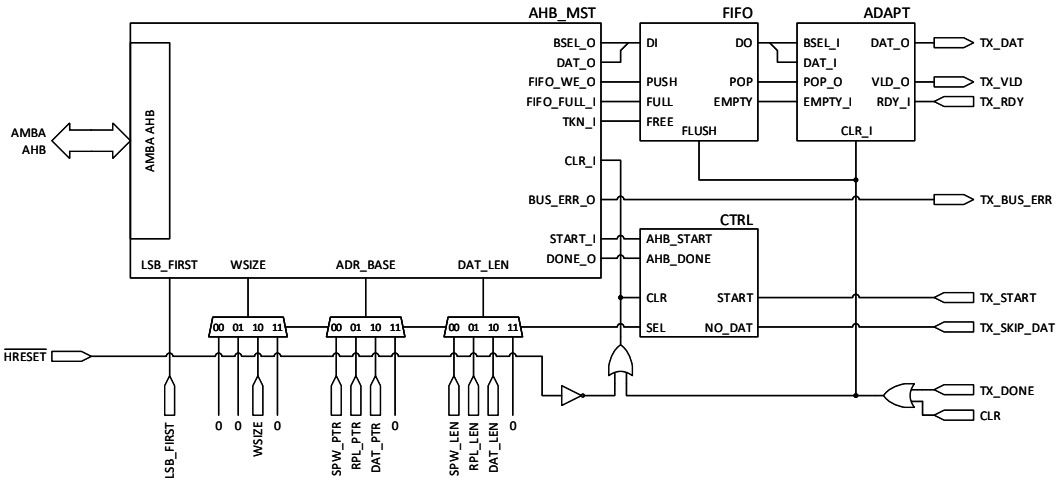
kde W_{ADR} je bitová šířka adresy paměti a funkce *ceil(x)* zaokrouhluje směrem nahoru na celá čísla. Ze vztahu výše vyplývá, že pro optimální využití paměti platí

$$W_{ADR} = \log_2 N_{REC}, \tag{5.3}$$

jinými slovy, počet buněk N musí být pro optimální implementaci takový, aby pokryl celý adresní prostor paměti.

5.1.6 Řadič DMA vysílače

Řadič DMA vysílače, označený TX_DMAC, umožňuje vysílači číst potřebná data z paměti nadřazeného systému. Jedná se konkrétně o SpaceWire adresy příkazu a odpovědi a přenášené bajty dat v případě operace zápisu nebo RMW. Schéma bloku TX_DMAC lze vidět na Obr. 5.8.



Obr. 5.8: Schéma řadiče DMA vysílače

Blok TX_DMAC se skládá z řadiče sběrnice, řídicího bloku spouštějícího řadič sběrnice a poskytujícího potřebná data tomuto řadiči, vyrovnávací paměť typu FIFO a nakonec tzv. adaptér, který přizpůsobuje data z řadiče sběrnice jednobajtovému datovému rozhraní mezi řadičem DMA a řadičem vysílače.

Řadič sběrnice obsluhuje protokol sběrnicového rozhraní. Pro navržené řadiče protokolu RMAP bylo jako rozhraní pro přístup do paměti zvoleno rozhraní AMBA AHB. Řadič sběrnice je tedy implementován jako AHB master. Podporuje transakce typu SINGLE a také dávkové transakce typu INCR. Typ transakce volí na základě počáteční adresy, šířky přenášeného slova a počtu přenášených bajtů. Pokud je šířka datového slova jeden bajt, vybere řídicí jednotka nejvhodnější kombinaci transakcí pro maximalizaci datové propustnosti, přičemž preferuje nejdelší možné dávkové transakce o největších možných datových šířkách. Během transakce předává čtená data do datového registru, z něhož jsou uložena přímo do vyrovnávací paměti. Ještě před uložením dat jsou však jednotlivé bajty řazeny podle endianity nadřazeného systému a podle hodnoty bitu LSB_FIRST specifikujícího pořadí odesílání bajtů z hlediska jejich významnosti. Aby nedošlo k nečekanému zaplnění vyrovnávací paměti, poskytuje tato paměť řadiči sběrnice informace o volném místu. Na základě této informace pak řadič sběrnice volí délku transakce, díky čemuž nemá potřebu transakce pozastavovat přenosem typu BUSY, a proto tento typ přenosu ani nepodporuje. Tím je dosaženo určitého zjednodušení tohoto řadiče. Řadič také podporuje možnost čtení ze stejné adresy. V tomto případě však používá pouze transakce typu SINGLE, jelikož dávkové transakce ze své podstaty musí adresu inkrementovat.

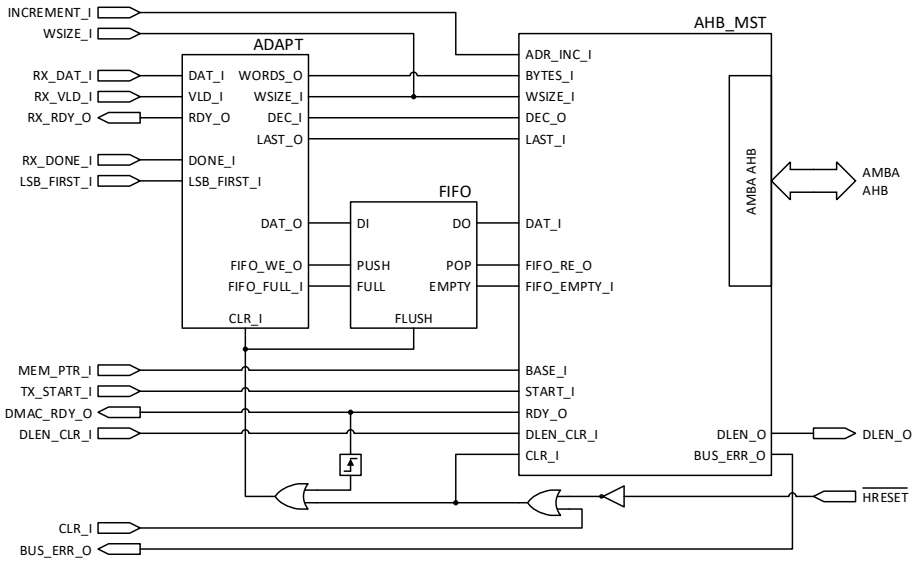
Při spuštění řadiče DMA spouští řídicí blok řadič sběrnice pro jednotlivé části paketu, přičemž potřebná data poskytuje řadiči sběrnice prostřednictvím multiplexorů. Řadič sběrnice provádí samotný přístup do paměti a získaná data uloží do vyrovnávací paměti spolu s bity indikujícími platné bajty. Adaptér pak čte data z vyrovnávací paměti a na základě zmíněných indikátorů platných bajtů předává bajt po bajtu získaná data řadiči přijímače.

5.1.7 Řadič DMA přijímače

Blok RX_DMAC slouží přijímači k ukládání dat přenášených v přijatých paketech odpovědi. Oproti řadiči DMA vysílače má jednodušší architekturu, kterou lze vidět na Obr. 5.9. Blok RX_DMAC se skládá z řadiče sběrnice AHB, vyrovnávací paměti a adaptéru, který je složitější než jeho protějšek na straně vysílače, a plní opačnou funkci, jinými slovy, uzpůsobuje jednobajtové datové rozhraní mezi řadičem DMA a řadičem přijímače pro rozhraní řadiče sběrnice.

Blok adaptéru během své činnosti ukládá přijaté datové bajty do registru, jehož šířka odpovídá té, kterou používá řadič sběrnice. Ukládání bajtů dat do tohoto registru probíhá na základě šířky datového slova a pořadí významnosti příchozích bajtů. To znamená, že přizpůsobení dat endianitě nadřazeného systému probíhá právě v tomto bloku. Pro zjednodušení adaptéru jsou pro menší šířky slova jednotlivé bajty replikovány tak, jak je znázorněno v Tab. 5.1. K uvedeným variantám přísluší následující kombinace pořadí významnosti a dané endianity:

- Varianta č. 1 – LE a LSB napřed, BE-32 a LSB napřed
- Varianta č. 2 – LE a MSB napřed, BE-32 a MSB napřed



Obr. 5.9: Řadič DMA přijímače

Adaptér obsahuje čítač počtu slov uložených do vyrovnávací paměti, jehož výstup je poskytnut řadiči sběrnice pro určení typu transakce. Pro řadič sběrnice je důležitá informace o tom, zda byla z řadiče přijímače přijata všechna data. Proto řadič přijímače indikuje poslední předaný bajt signálem, který adaptér používá k signalizaci řadiči sběrnice, že do vyrovnávací paměti byla uložena poslední data.

Tab. 5.1: Princip replikace datových bajtů pro 64bitovou datovou sběrnici

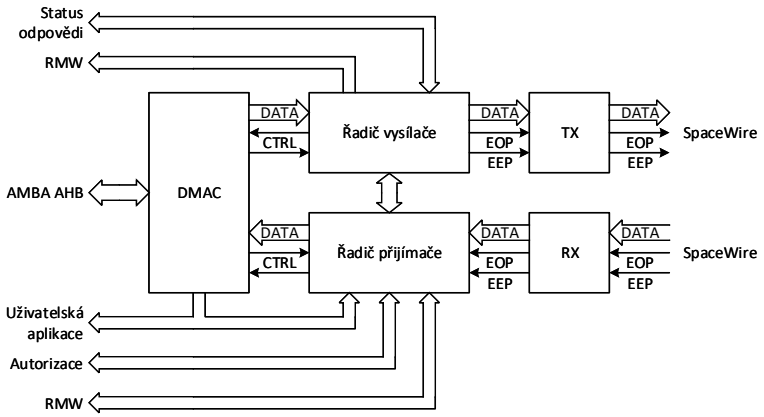
Šířka slova [b]	Výstup (varianta č. 1)								Výstup (varianta č. 2)							
8	B0	B0	B0	B0	B0	B0	B0	B0	B0	B0	B0	B0	B0	B0	B0	B0
16	B1	B0	B1	B0	B1	B0	B1	B0	B0	B1	B0	B1	B0	B1	B0	B1
32	B3	B2	B1	B0	B3	B2	B1	B0	B0	B1	B2	B3	B0	B1	B2	B3
64	B7	B6	B5	B4	B3	B2	B1	B0	B0	B1	B2	B3	B4	B5	B6	B7

Řadič sběrnice podporuje oproti svému protějšku z vysílače funkci zápisu a používá pouze fixní šířku přenášeného slova. Dále poskytuje uživatelské aplikaci údaj o počtu bajtů skutečně zapsaných do paměti. Tato hodnota totiž nemusí odpovídat počtu bajtů uvedeném v hlavičce paketu.

Údaje o šířce slova a pořadí významnosti přijímaných bajtů jsou, stejně jako bázová adresa a její inkrementace, dány uživatelskou aplikací při odeslání příkazu, a tudíž uloženy, kromě jiných údajů, v registru transakcí.

5.2 Návrh řadiče cílového uzlu

Řadič cílového uzlu z hlediska architektury vychází z řadiče iniciátoru, viz Obr. 5.10. Výrazným rozdílem je absence registru transakcí, který má význam pouze pro iniciátor. Dalším rozdílem je sdílený řadič DMA pro přijímač i vysílač. Cílový uzel totiž nejdříve zpracuje příchozí příkaz a až poté odesílá odpověď, není proto nutné, aby obě části měly dedikovaný řadič DMA.



Obr. 5.10: Architektura cílového uzlu protokolu RMAP

Bloky RX a TX jsou stejné, jako v případě iniciátoru. Blok řadiče DMA byl odvozen z řadičů DMA iniciátoru a uzpůsoben tak, že řadič vysílače má přístup ke čtení z paměti a řadič přijímače může do paměti pouze zapisovat. Díky tomu bylo možno řadiče vysílače a přijímače taktéž odvodit z řadičů v iniciátoru.

V případě iniciátoru byla data jednosměrně předávána vysílačem přijímači prostřednictvím registru transakcí. V případě cílového uzlu řadič přijímače působí jako hlavní řídicí blok a v případě potřeby spouští řadič vysílače. Jedná se konkrétně o odeslání paketu odpovědi a o požadavek na provedení operace RMW. Řadič přijímače také poskytuje řadiči vysílače informace o přijatém příkazu a o zjištěných chybách pro generaci odpovědi.

Dalšími rozdílem je přítomnost rozhraní pro autorizaci, kontrolní vyrovnávací paměť, pro RMW operaci a pro generátor stavového slova. Z důvodu flexibility jsou tyto funkce řešeny pomocí externích bloků, které lze pak přizpůsobit příslušné aplikaci.

5.2.1 Řadič přijímače

Řadič přijímače je založen na svém protějšku z iniciátoru, je ale podstatně komplexnější, viz Příloha G. Stále používá registrový blok pro ukládání dat hlavičky příkazu a jejich kontrolu, stavový automat jako hlavní řídicí jednotku, datový čítač a registry pro předání dat řadiči DMA. Přibyl však čítač pro počítání bajtů adresy odpovědi a vyrovnávací paměť pro adresu odpovědi, jejíž výstupní rozhraní je přivedeno přímo do řadiče vysílače. K této paměti také patří filtrační blok, který slouží k odstranění počátečních nul v adrese odpovědi, jelikož ty nejsou v paketu odpovědi přenášeny. Výjimkou je případ, kdy jsou všechny bajty adresy odpovědi nulové – v tom případě je uložen jeden bajt s nulovou hodnotou.

Výstupní část je rozšířena tak, aby podporovala přenos dat do řadiče DMA z paketu, kontrolní paměti, anebo z registru obsahujícího výsledek operace RMW. S tím souvisí také nové rozhraní pro kontrolní paměť, do které jsou ukládána data, aby mohlo být v případě chyby CRC dat zabráněno jejich zápisu do paměti. Tato funkce je vyhrazena pro operaci zápisu s kontrolou integrity dat. Jako kontrolní paměť je předpokládána paměť typu FIFO, a tomu odpovídá také zmíněné rozhraní.

Dalším novým prvkem je rozhraní pro autorizaci, které obsahuje všechny údaje o paketu specifikované v [4], a dále signál pro požadavek na autorizaci a signál indikující její dokončení. Dále k tomuto rozhraní patří tři chybové signály, které indikují neplatnou logickou adresu cílového uzlu, neplatný klíč a zamítnutí paketu z jakéhokoliv jiného důvodu. Tyto chybové signály jsou platné při shodě signálů požadavku a dokončení autorizace.

Pro implementaci operace RMW bylo vyhrazeno další rozhraní, které je rozděleno mezi řadič vysílače a řadič přijímače. Řadič přijímače poskytuje data a masku získané z paketu. Operace RMW je zahájena tak, že řadič přijímače odešle požadavek řadiči vysílače, který posléze načte původní slovo z paměti a nastaví požadavek na operaci RMW. Jakmile jsou nová data dostupná, je řadiči přijímače signalizována platnost výsledku operace RMW a toto výsledné slovo je v dalším hodinovém taktu uloženo do registru.

Datový čítač byl modifikován tak, že uchovává délku dat přečtenou z paketu, jelikož může být během činnosti řadiče přijímače několikrát inicializován. Příkladem může být ukládání dat do kontrolní paměti a jejich pozdější přesun do řadiče DMA. Čítač také obsahuje vstup pro indikaci RMW operace, jelikož délka dat v hlavičce paketu v případě operace RMW zahrnuje také počet bajtů masky. Dále má výstup pro číslo přenášeného bajtu a kontrolu správné délky dat, obojí pro operaci RMW.

Při přijetí nového paketu řadič postupně načítá data hlavičky a kontroluje její CRC. Pokud je zaregistrována chyba hlavičky, je činnost v principu stejná jako v případě iniciátoru s tím rozdílem, že může být za určitých okolností odeslána odpověď zpět iniciátoru. V opačném případě následuje autorizace a v případě chyby autorizace je postup totožný s postupem uvedeným výše. Je-li autorizace úspěšná, pak záleží na typu operace.

Pro operaci zápisu s kontrolou dat jsou data ukládána do kontrolní paměti. Pokud dojde k chybě CRC dat, nečekanému zaplnění kontrolní paměti nebo k nečekaně nízkému počtu datových bajtů, je zaregistrována příslušná chyba, zbytek paketu je případně zahozen a je-li to požadováno, je odeslána odpověď. Pokud vše proběhne v pořádku, následuje přenos dat z kontrolní paměti do řadiče DMA a případné odeslání paketu odpovědi.

V případě operace RMW jsou postupně načteny registry pro data a masku a pokud je datová část paketu kratší, než bylo očekáváno, je postupováno stejně jako v případě uvedeném výše. V opačném případě je vyžádána operace RMW a po jejím dokončení je její výsledek zapsán zpět do paměti. Následně je odeslána odpověď.

V případě operace zápisu bez kontroly dat a operace čtení je přikročeno rovnou k přístupu k paměti. Pro operaci čtení je se o čtení příslušných dat a generaci a odeslání paketu odpovědi postará řadič vysílače. V případě operace zápisu bez kontroly dat jsou data přenesena do paměti pomocí řadiče DMA a poté je provedena kontrola CRC dat. Pokud je požadována odpověď, je odeslána pomocí řadiče vysílače. Pokud jsou zapisovaná data předčasně ukončena zakončovacím znakem, počká řadič přijímače nejprve na ukončení činnosti řadiče DMA.

Během ukládání dat do paměti může řadič DMA indikovat chybu. V takovém případě je ukončeno čekání na řadič DMA a je zahájena zakončovací sekvence, která v závislosti na typu operace může zahrnovat zahození zbytku paketu a odeslání paketu odpovědi.

5.2.2 Řadič vysílače

Řadič vysílače je podstatně jednodušší oproti řadiči přijímače a jeho schéma lze nalézt v Příloze F. Obsahuje výstupní registry pro předání dat, zakončovacích znaků a řídicích signálů pro jednotku CRC bloku TX. Dále obsahuje posuvný registr pro hlavičku odpovědi a čítač odpočítávající jednotlivé bajty hlavičky, čítač datových bajtů, registr pro uchování chyby sběrnice a posuvný registr pro uložení dat vyčtených z paměti pro operaci RMW.

Tento posuvný registr je načten pomocí řadiče DMA a ukládání bajtů do jednotlivých částí registru je určeno datovým čítačem. Řadič vysílače zachází s bajty přijatými z řadiče DMA jako s bajty klesající významnosti, tzn. bajt MSB je z řadiče DMA přijímán jako první. Registr má paralelní výstup pro RMW rozhraní a výstup o šířce jednoho bajtu pro postupné odeslání dat v paketu odpovědi.

Výstupní datový registr má na svém vstupu multiplexor, pomocí kterého řadič volí zdroj dat. Stejným způsobem přepíná zdroj signálu indikujícího platná data pro výstup TX_VLD_O. Poslední částí je stavový automat, který řídí činnost řadiče.

Řadič vysílače pracuje ve dvou režimech. Prvním je režim operace RMW, který probíhá tak, že jsou prvně vyčtena data z paměti a uložena do vyhrazeného posuvného registru, a poté je požádáno o operaci RMW. Jakmile je signalizován konec RMW operace, přejde automat do stavu nečinnosti. V případě chyby na sběrnici jsou vyčtena dostupná data a následně je řadiči přijímače indikována chyba operace RMW.

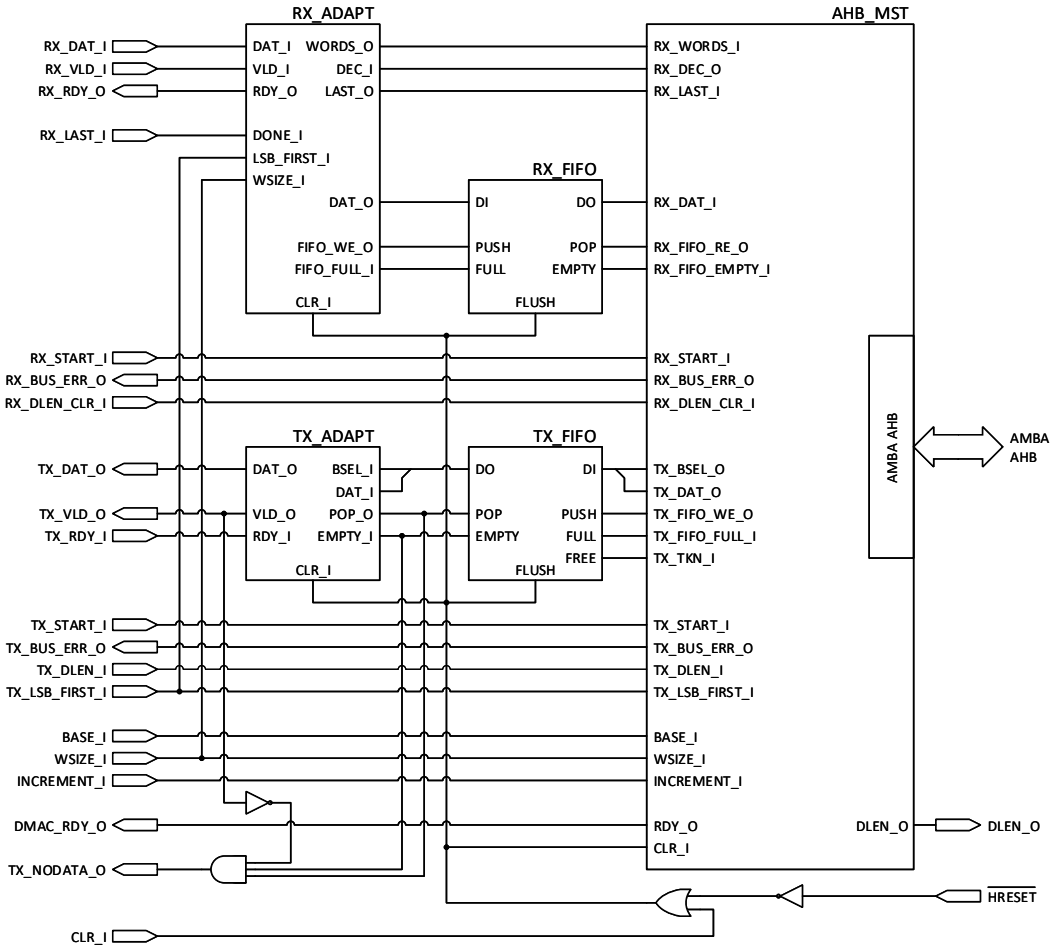
Druhým režimem je odeslání odpovědi, při kterém je nejprve vyžádána generace stavového slova pro paket odpovědi. Po jeho získání je z fronty v řadiči přijímače postupně vyčítána a odesílána adresa odpovědi, pokud byla součástí paketu příkazu. Následuje postupné odesílání bajtů hlavičky uložených v posuvném registru, vřazení kontrolního součtu a nulování jednotky CRC. Pokud se jedná o operaci zápisu, je za kontrolní součet zařazen znak EOP. Pokud se jedná o operaci RMW, jsou nejprve odeslána data z posuvného registru pro operaci RMW, následně je do paketu zařazen kontrolní součet dat, jednotka CRC je nulována a znak EOP je zařazen na konec paketu. V případě operace čtení jsou pomocí řadiče DMA čtena data z paměti a postupně odesílána s následným vřazením kontrolního součtu, nulováním jednotky CRC a zakončením paketu znakem EOP.

Pokud během přístupu do paměti dojde k chybě, jsou odeslána dostupná data a je zařazen znak EEP bez kontrolního součtu. To platí pro operace čtení a RMW.

5.2.3 Řadič DMA

Jak již bylo zmíněno, vysílač i přijímač sdílí jeden řadič DMA. Tento řadič je odvozen z obou řadičů DMA iniciátoru a v základu se jedná o jejich kombinaci s tím, že řadič přijímače má k dispozici pouze port pro zápis a řadič vysílače pouze port pro čtení.

Schéma řadiče DMA lze vidět na Obr. 5.11. Z iniciátoru byly použity oba bloky adaptéru a také vyrovnávací paměti. Řadič sběrnice vznikl vhodným sloučením částí obou řadičů sběrnice iniciátoru. Hlavní bloky byly převzaty a použitím dodatečné logiky a úpravou řídicí jednotky bylo dosaženo kýženého výsledku. Při zápisu dat do paměti lze použít výstup DLEN_O jako indikátor počtu zapsaných bajtů. Při čtení dat z paměti čítá datový čítač v řadiči sběrnice směrem dolů, takže hodnota na výstupu DLEN_O ztrácí v takovém případě smysl.



Obr. 5.11: Řadič DMA cílového uzlu

6 OVĚŘENÍ FUNKCE A IMPLEMENTACE

Navržené řadiče protokolu RMAP bylo nutno ověřit z hlediska jejich správné funkce. Ověření se skládalo ze simulací, které se skládaly z cílených testů a testů s náhodně generovanými stimuly pro tyto řadiče, a dále z testování navržených řadičů v cílovém obvodu FPGA.

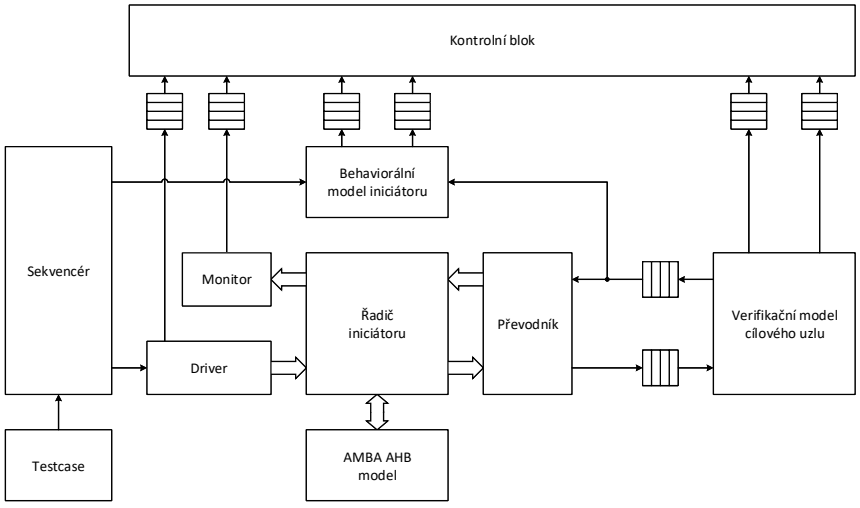
6.1 Cílené testy

Jedná se o ručně psané testy v jazyce VHDL, které slouží k ladění navržených řadičů. Princip těchto testů se liší podle typu testovaného řadiče. V případě iniciátoru byly nastavováním příslušných vstupů iniciovány požadavky na odeslání paketu příkazu a bylo sledováno chování iniciátoru z hlediska dat v odeslaném paketu, hlášení chyb a přenosu dat po sběrnici AMBA AHB. Poté byl vytvořen příslušný paket odpovědi a odeslán zpět iniciátoru. Obdobně jako u příkazu bylo sledováno uživatelské rozhraní a přenos dat do paměti pomocí řadičů DMA. V případě testování cílového uzlu byly testovanému řadiči poslány pakety příkazu a byly sledovány pakety odpovědi a chování testovaného řadiče z hlediska zpracování paketu příkazu, přístupu řadiče DMA do paměti a komunikace s uživatelskou aplikací.

Tyto testy byly provedeny jako první, aby byly zjištěny a opraveny ty nejzásadnější chyby. V dalších fázích testování byly tyto testy použity pro identifikaci a opravu dalších zjištěných chyb.

6.2 Náhodně generované testy

Po základním ověření navržených řadičů byly provedeny komplexní simulace. Tyto simulace byly navrženy jako automatizované s využitím náhodného generování vstupních podnětů. Proces náhodné generace byl řízen pomocí konfigurovatelných omezujících podmínek. V těchto simulacích byly použity navržené verifikačních modely iniciátoru a cílového uzlu a bylo využito jazyka SystemVerilog.



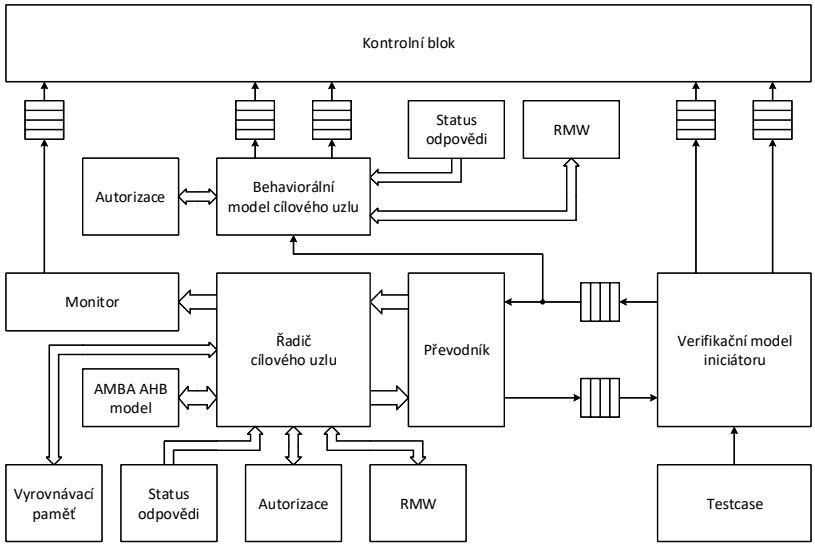
Obr. 6.1: Verifikační prostředí pro testování řadiče iniciátoru

Na Obr. 6.1 lze vidět architekturu testovacího prostředí pro řadič iniciátoru. Skládá se ze sekvencéru, který generuje parametry požadovaného příkazu. Tyto parametry jsou pak předány bloku Driver, který reprezentuje rozhraní mezi abstraktní formou dat generovaných sekvencérem a uživatelským portem řadiče iniciátoru. Tento blok převezme parametry požadavku a požadavek předá řadiči iniciátoru. Poté čeká, než iniciátor poskytne odpověď, zaregistruje případné chyby a tyto údaje předá kontrolnímu bloku. K testovanému řadiči je připojen model sběrnicového subsystému AMBA AHB, který krom sběrnice obsahuje dekodér adres, arbitr, výchozí master, výchozí slave a slave s pamětí, do které iniciátor může přistupovat. Tento blok tak modeluje chování paměťového rozhraní. K iniciátoru je také připojen verifikační model cílového uzlu, který byl již popsán v kapitole 4.2. Propojení je realizováno pomocí bloku, který převádí signály testovaného iniciátoru do abstraktní formy, kterou používá verifikační model, a zpět. Blok Monitor sleduje přijímací část uživatelského rozhraní řadiče iniciátoru a přebírá od něj data o přijatém paketu a o chybách, která pak odesílá kontrolnímu bloku.

Pro snazší kontrolu funkce navržených řadičů byly vytvořeny behaviorální modely těchto řadičů. Tyto modely používají model paměti, který je částí modelu aplikace cílového uzlu popsané v kapitole 4.2. Aby testovaný řadič a příslušný behaviorální model obdržely stejné příchozí pakety, je do přijímací části testovacího prostředí z pohledu testovaného řadiče vřazen blok, který pakety duplikuje a odesílá oběma blokům.

Informace pro kontrolu jsou pro kontrolní blok ukládány do schránek. Ten cyklicky tyto informace sbírá a kontroluje průběh komunikace. Porovnává údaje z testovaného řadiče a jeho behaviorálního modelu, zejména chybové bity a obsah pamětí, dále pak kontroluje přijatý paket příkazu vůči obsahu paměti iniciátoru, přijatý paket odpovědi vůči obsahu paměti cílového uzlu a chyby hlášené verifikačním modelem. Kvůli možnému přepisu obsahů pamětí před jejich kontrolou, je součástí údajů uložených ve schránkách také odpovídající obsah příslušné paměti, který byl zkopírován ve vhodném okamžiku.

Obr. 6.2 znázorňuje testovací prostředí pro testování řadiče cílového uzlu. Architektura tohoto prostředí je založena na testovacím prostředí iniciátoru, byla však upravena pro testování cílového uzlu. Není zde již použit sekvencér, jelikož verifikační model iniciátoru příkazy generuje sám. Stejně tak není přítomen blok Driver, jelikož uživatelské rozhraní řadiče cílového uzlu pouze poskytuje údaje o příkazu, k čemuž stačí blok Monitor. K testovanému řadiči a jeho behaviorálnímu modelu jsou připojeny bloky pro autorizaci, operaci RMW a generaci stavového slova pro paket odpovědi. Oproti testovanému řadiči však jeho behaviorální model má vyrovnávací paměť pro kontrolu integrity dat integrovanou a není tak třeba externího modelu.

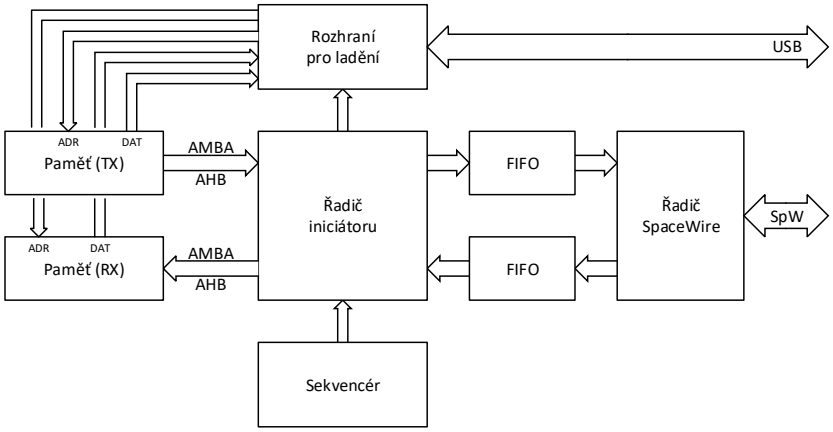


Obr. 6.2: Verifikační prostředí pro testování řadiče cílového uzlu

Konfigurace testů je v obou případech prováděna pomocí bloku Testcase, který konfiguruje parametry generace příkazů v sekvencéru v případě testování iniciátoru a v modelu iniciátoru v případě testování cílového uzlu.

6.3 Testování v obvodu FPGA

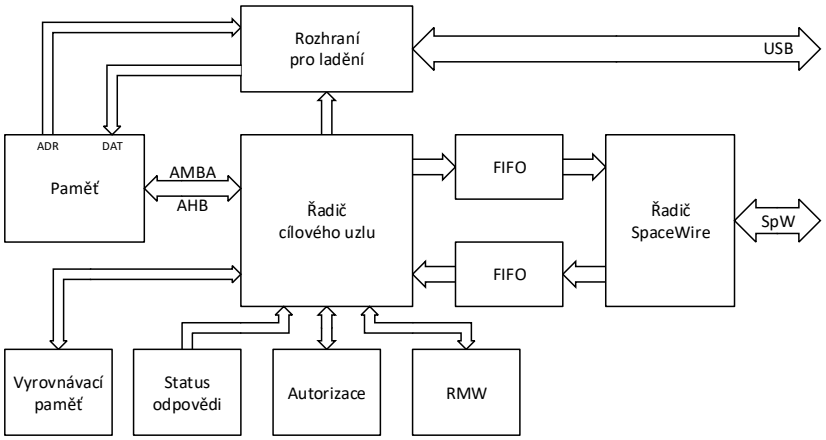
Poslední fází této práce bylo otestování vytvořených řadičů v obvodu FPGA. Tímto obvodem byl konkrétně obvod Spartan-3E XC3S1600E. Navržená jádra řadičů byla integrována do testovacích obvodů, které byly pro každý řadič specifické.



Obr. 6.3: Testovací obvod pro řadič iniciátoru

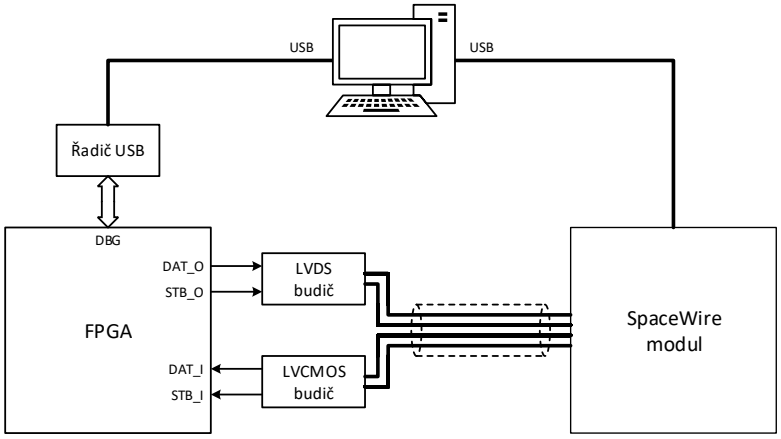
Blokové schéma testovacího obvodu pro řadič iniciátoru je znázorněno na Obr. 6.3. Součástí tohoto obvodu je krom testovaného řadiče sekvencér, který generuje požadavky na odeslání příkazu, dále pak paměti typu FIFO mezi testovaným řadičem a řadičem sítě SpaceWire použitým k testování. Pro tento testovací obvod byl vytvořen jednoduchý obvod typu slave s dvouportovou pamětí, kde jeden z portů poskytuje rozhraní AMBA AHB. Pro jednoduchost testovacího obvodu byly použity dvě odlišné instance

této paměti, jedna pro vysílač a druhá pro přijímač. Díky tomu bylo možno připojit testovaný řadič k pamětem přímo. Dále byl použit modul pro komunikaci s externím řadičem USB, ke kterému byly připojeny některé porty uživatelské aplikace, zejména chybové porty, a druhé porty pamětí. Tento modul byl využit pro případné ladění testovaného obvodu.



Obr. 6.4: Testovací obvod pro řadič cílového uzlu

Na Obr. 6.4 lze vidět rozvržení testovacího obvodu pro testování cílového uzlu. Od předešlého obvodu se liší tím, že místo sekvencéru používá bloky pro autorizaci, operaci RMW a generaci stavového slova paketu odpovědi. Vzhledem k tomu, že řadič cílového uzlu obsahuje pouze jedno rozhraní pro přístup k paměti, byla použita jen jedna instance paměťového bloku.



Obr. 6.5: Testovací prostředí

Obr. 6.5 znázorňuje testovací prostředí. Obvod FPGA s testovaným řadičem je připojen k počítači prostřednictvím již zmíněného řadiče USB pro případné ladění. Dále je připojen k budičům pro standard LVDS a pomocí kabelu popsaného již v kapitole 1.1 propojen s referenčním modulem SpaceWire, který slouží jako referenční obvod pro komunikaci. Tento modul je připojen k počítači přes rozhraní USB a pomocí něj byly přijímány a odesílány příslušné pakety protokolu RMAP. Byly otestovány operace zápisu, čtení i RMW a testované řadiče pracovaly podle očekávání.

6.4 Implementace

Jak již bylo uvedeno, navržené řadiče byly implementovány pro obvod XC3S1600E. Tento model má k dispozici nejvíce programovatelné logiky v rodině obvodů FPGA Spartan-3E. Jako poslední část práce byly analyzovány výsledky syntézy navržených řadičů z hlediska jejich velikosti a maximální frekvence.

Pro určení množství použitých zdrojů obvodu FPGA byla provedena syntéza každého řadiče zvlášť. Nastavení parametrů řadičů je uvedeno v Tab. 6.1.

Tab. 6.1: Parametry řadičů

Společná nastavení		**Nastavení iniciátoru**	
Parametr	**Nastavení**	**Parametr**	**Nastavení**
Kapacita FIFO	16 slov	Kapacita registru transakcí	1 transakce
Implementace pamětí	LUT		16 transakcí
Implementace jednotky CRC	paralelní		32 transakcí
Šířka datové sběrnice AHB	32 bitů		64 transakcí

Pro určení odhadu maximální frekvence byl každý řadič integrován do bloku, který všechny porty řadiče připojil k registrům, čímž byl odstraněn vliv kombinačních cest na odhad maximální frekvence. Výsledky této analýzy jsou uvedeny v Tab. 6.2.

Tab. 6.2: Odhad využití zdrojů FPGA a maximální frekvence

Iniciátor				
Kapacita registru transakcí	**1 transakce**		**16 transakcí**	
Druh zdrojů obvodu FPGA	LUT	Registry	LUT	Registry
Dostupné zdroje [-]	29504	29504	29504	29504
Využití [-]	2336	727	2733	989
Využití [%]	7	2	9	3
Maximální frekvence [MHz]	97,663		85,325	
Kapacita registru transakcí	**32 transakcí**		**64 transakcí**	
Druh zdrojů obvodu FPGA	LUT	Registry	LUT	Registry
Dostupné zdroje [-]	29504	29504	29504	29504
Využití [-]	3268	1309	4388	1899
Využití [%]	11	4	14	6
Maximální frekvence [MHz]	71,347		69,075	
Cílový uzel				
Druh zdrojů obvodu FPGA	LUT		Registry	
Dostupné zdroje [-]	29504		29504	
Využití [-]	2283		851	
Využití [%]	7		2	
Maximální frekvence [MHz]	88,998			

7 ZÁVĚR

Cílem této práce bylo navrhnout, implementovat a verifikovat řadiče protokolu RMAP, který slouží ke vzdálenému přístupu do paměti koncových uzlů sítě SpaceWire.

V rámci rešerše této práce byly prostudovány standardy sítě SpaceWire a na ní založeného protokolu RMAP. V rámci sítě SpaceWire byly uvedeny základní informace o této síti, konkrétně o fyzické vrstvě vzhledem k přenosovému médiu a způsobu kódování dat a hodinového signálu, linkové vrstvě a topologii sítě včetně způsobu směrování paketů v síti SpaceWire. V rámci studia protokolu RMAP byly prostudovány struktury paketů příkazu a odpovědi pro jednotlivé druhy operací a procesy zpracování těchto paketů iniciátorem a cílovým uzlem. Dále byly podle standardu definovány možné chyby paketů a způsoby reakce přijímacího modulu. Jako poslední část rešerše byl prostudován standard sběrnicového rozhraní AMBA AHB, jelikož realizované řadiče protokolu RMAP pro přístup do paměti používají právě toto rozhraní.

V praktické části práce byly nejprve vytvořeny verifikační modely iniciátoru a cílového uzlu protokolu RMAP pomocí jazyka SystemVerilog. Tyto modely slouží jako komunikační modely pro verifikaci realizovaných řadičů. Samotné řadiče byly navrženy a realizovány pomocí jazyka VHDL a umožňují uživateli měnit jejich parametry podle potřeb příslušné aplikace. Těmito parametry jsou druh implementace pamětí a jejich kapacita, druh implementace jednotek CRC anebo kapacita registru transakcí v případě řadiče iniciátoru. Pro rozhraní AMBA AHB lze volit šířku datové sběrnice v rozsahu 8, 16, 32 a 64 bitů a také endianitu nadřazeného systému.

Pro ověření funkce řadičů pomocí simulace bylo použito cílených testů a automatizovaných testů s náhodnou generací stimulů a konfigurovatelnými omezeními generace. Cílené testy byly vytvořeny pomocí jazyka VHDL, pro automatizované testy bylo využito jazyka SystemVerilog. Během automatizovaných testů bylo pro každý řadič vygenerováno přibližně 10 000 paketů. Po ověření funkce pomocí simulací proběhlo testování řadičů v obvodu FPGA, kde byla druhá strana komunikace realizována použitím referenčního modulu pro síť SpaceWire. Pomocí tohoto modulu byly odesílány a přijímány pakety protokolu RMAP. Byla ověřena funkce všech tří základních typů operací včetně ověření obsahu pamětí používaných testovanými řadiči.

Poslední částí práce byla analýza maximální frekvence řadičů a jejich využití zdrojů obvodu FPGA. Na základě výsledků analýzy řadiče iniciátoru lze stanovit, že jeho registr transakcí ve velké míře přispívá k růstu využití zdrojů. Hlavním důvodem je velikost buňky tohoto registru, která obsahuje 17 klopných obvodů a krom několika hradel také dva 16bitové komparátory. Tento registr také ovlivňuje maximální frekvenci iniciátoru, která se v použitém rozmezí konfigurace pohybuje mezi 60 a 100 MHz. Maximální frekvence cílového uzlu se pohybuje kolem 89 MHz a vyžaduje přibližně stejné množství zdrojů FPGA jako řadič iniciátoru s minimální kapacitou registru transakcí. Je však nutno podotknout, že tyto výsledky jsou výstupem syntézy, a proto se jedná o pouhý odhad.

LITERATURA

[1] Introduction. *SpaceWire* [online]. Francie: European Space Agency, ©2000-2006 [cit. 2016-11-26]. Dostupné z: http://spacewire.esa.int/content/Home/HomeIntro.php

[2] ECSS-E-ST-50-12C. *SpaceWire: Links, nodes, routers and networks*. 2nd issue. Noordwijk (Nizozemsko): ESA Requirements and Standards Division, 2008.

[3] ECSS-E-ST-50-5x series. *SpaceWire* [online]. Francie: European Space Agency, ©2000-2006 [cit. 2016-11-26]. Dostupné z: http://spacewire.esa.int/content/Standard/Draft_ECSS-E50-11.php

[4] ECSS-E-ST-50-52C. *SpaceWire: Remote memory access protocol*. Noordwijk (Nizozemsko): ESA Requirements and Standards Division, 2010.

[5] ARM IHI 0011A. *AMBA™ Specification*. Rev 2.0. Cambridge (Spojené království): ARM Limited, 1999, 230 s. Dostupné z: http://infocenter.arm.com/help/index.jsp

[6] ARM DDI 0432C. *Cortex-M0: Technical Reference Manual*. Revision r0p0, issue C. Cambridge (Spojené království): ARM Limited, 2009, 68 s. Dostupné také z: http://infocenter.arm.com/help/index.jsp

[7] ADIGA, Harsha. Writing endian-independent code in C: Don't let endianness "byte" you. *IBM developerWorks* [online]. Armonk, New York (Spojené státy americké): IBM Corporation, 2007, 15. 5. 2007 [cit. 2017-05-16]. Dostupné z: https://www.ibm.com/developerworks/aix/library/au-endianc/index.html

[8] Byte Ordering. *Apple Developer* [online]. Cupertino, Kalifornie (Spojené státy americké): Apple Incorporated, 2009, 21. 10. 2009 [cit. 2017-05-16]. Dostupné z: https://developer.apple.com/library/content/documentation/CoreFoundation/Conceptual/CFMemoryMgmt/Concepts/ByteOrdering.html

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

DAT_I	vstupní data Data-Strobe enkodéru
CLK_I	vstupní hodinový signál Data-Strobe enkodéru
DAT_O	výstupní data Data-Strobe enkodéru
STB_O	výstupní Strobe signál Data-Strobe enkodéru
g(x)	polynom pro výpočet CRC
W_{DAT}	šířka datové sběrnice
N_{MST}	počet obvodů typu master
N_{SLV}	počet obvodů typu slave
N_{REC}	počet záznamů, které lze uložit do registru transakcí iniciátoru
N_R	počet regionů v registru transakcí
N_B	počet bloků v registru transakcí
N_C	počet buněk v registru transakcí
W_{ADR}	šířka adresové sběrnice paměti v registru transakcí
ceil(x)	funkce zaokrouhlení na celá čísla směrem nahoru

CRC	Cyclic Redundancy Check – cyklická redundantní kontrola
EEP	Error End of Packet – chybný konec paketu
EOP	End of Packet – konec paketu
ESA	European Space Agency – Evropská vesmírná agentura
FPGA	Field Programmable Gate Array – programovatelné hradlové pole
GPS	Global Positioning System – globální lokalizační systém
HIC	Heterogenous Interconnect – heterogenní rozhraní
IEEE	Institute of Electrical and Electronics Engineers – Institut pro elektrotechnické a elektronické inženýrství
LFSR	Linear Feedback Shift Register – posuvný registr s lineární zpětnou vazbou
LSB	Least Significant Bit/Byte – nejméně významný bit/bajt
LVDS	Low-Voltage Differential Signaling – nízkonapěťový diferenciální signál
MSB	Most Significant Bit/Byte – nejvíce významný bit/ bajt

PECL	Positive Emitter-Coupled Logic – kladně polarizovaná emitorově vázaná logika
RMAP	Remote Memory Access Protocol – protokol pro vzdálený přístup do paměti
RMW	Read-Modify-Write – operace čtení, modifikace a zpětného zápisu
VHDL	VHSIC Hardware Description Language – popisný jazyk pro modelování a návrh číslicových systémů
AMBA	Advanced Microcontroller Bus Architecture – specifikace sběrnicových rozhraní pro použití v integrovaných obvodech
ARM	Advanced RISC Machines – společnost zabývající se návrhem architektur mikroprocesorů a jejich sběrnicových rozhraní
AHB	Advanced High-performance Bus – pokročilá vysokorychlostní sběrnice
LE	Little Endian – řazení dat v paměti od nejnižší významnosti
BE	Big Endian – řazení dat v paměti od nejvyšší významnosti
DMA	Direct Memory Access – přímý přístup do paměti
RAM	Random Access Memory – paměť s náhodným přístupem
LUT	Look Up Table – náhledová tabulka, základní stavební blok obvodů FPGA
FIFO	First In, First Out – paměť typu fronta

SEZNAM PŘÍLOH

A Chyby paketů a reakce RMAP modulů pro operaci zápisu i
B Chyby paketů a reakce RMAP modulů pro operaci čtení iii
C Chyby paketů a reakce RMAP modulů pro operaci RMW iv
D Řadič vysílače iniciátoru vi
E Řadič přijímače iniciátoru vii
F Řadič vysílače cílového uzlu viii
G Řadič přijímače cílového uzlu ix